Software Engineering Institute

Considerations for Using Agile in DoD Acquisition

Mary Ann Lapham
Ray Williams
Charles (Bud) Hammons
Daniel Burton
Alfred Schenker

April 2010

TECHNICAL NOTE
CMU/SEI-2010-TN-002

Acquisition Support Program – Air Force

Carnegie Mellon

Table of Contents

List of Figures

List of Tables

Acknowledgments

The authors would like to express our appreciation to all those who took time out of their busy schedules to allow us to interview them. Special thanks go to the Joint Mission Planning System (JMPS) Program Management Office (PMO) and BAE Systems staff. Thank you to all the Carnegie Mellon® Software Engineering Institute (SEI) colleagues who provided articles, blogs, and encouragement.

To all those who tolerated endless questions during the 2009 Agile Development Practices Conference, many thanks as you were all gracious and so willing to share your knowledge and excitement about Agile concepts and application.

To our reviewers—your thoughtful and precise insights added great value, which we greatly appreciated. We extend our sincerest thanks to the following people:

- Jim Highsmith, Signer of Agile Manifesto
- Linda Rising, Independent Agile Consultant
- Sean Mullen, Mitre
- Jim Corbin, BAE Systems, Geospatial Solutions Engineering Director
- Lt Col Daniel Ward, USAF
- Dr. Doug Buettner, Aerospace
- Joe Tatem, Raytheon
- Stephany Bellomo, SEI
- Nanette Brown, SEI
- John Foreman, SEI
- Dr. John Goodenough, SEI
- Harry Levinson, SEI
- Dr. Robert Nord, SEI
- Ipek Ozkaya, SEI

® Carnegie Mellon is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

Abstract

This report explores the questions: *Can Agile be used in the DoD environment? If so, how?* Lessons learned from actual DoD programs that have employed and are employing Agile are provided as well as information gleaned from the myriad articles and books available on Agile. While this report does not pretend to cover every paper or thought published about Agile in the DoD world, it provides an overview of some challenges in using Agile; an overview of how some programs have addressed these challenges; and some additional recommendations on dealing with these challenges. The intended audience is policy makers, program office staff, and software development contractors who are contemplating proposing the use of Agile software development methods.

It is the hope of the authors that this paper stimulates discussion about and appropriate adoption of Agile in the DoD world. We hope to obtain further data so that our list of considerations can be updated and expanded for use by all practitioners.

Organization of This Report

This report is organized as follows:

Executive Summary (page xiii) contains highlights of this report, specifically Agile methods and their use in the DoD.

Overview (page 1) describes the approach taken to develop this report, what is included and what is excluded.

What is Agile? (page 3) provides a definition/history of Agile, a generic Agile example, and a short comparison of Agile to Waterfall.

Interview Observations and Findings (page 11) presents the results from interviewing specific programs and includes pitfalls, issues, and potential solutions.

DoD 5000 Series and Agile—Potential Issues and Conflicts (page 23) details the results of an analysis of the DoD 5000 Series and how it impacts Agile use within the DoD.

Considerations for Applying Agile in the DoD (page 35) provides multiple considerations for applying Agile in the DoD. It also includes discussions of various Agile concepts and addresses potential hurdles for implementing them within DoD.

Conclusion (page 44) contains a summary of this report and suggestions for future research on using Agile within the DoD.

Appendix A (page 46) provides a variety of Agile methods and their definitions.

Appendix B (page 49) identifies and debunks common objections to using Agile.

Appendix C (page 51) details areas of concern and their considerations for using Agile in the DoD.

Appendix D (page 54) defines acronyms used throughout this report.

Appendix E (page 56) contains the text of the “FIST Manifesto.”

Executive Summary

In 2009 the SEI was tasked by Mr. Blaise Durante, Air Force Deputy Assistant Secretary for Acquisition Integration (SAF/AQX), to assess the state of the practice of Agile development in government software acquisitions. This team was assembled to complete that assessment.

This report is the result of this assessment, and is meant to debunk the prevalent myth that Agile and Department of Defense (DoD) practices are incompatible. Our focus is on the software development arena, basing our information on actual acquisition experience and a sampling of the relevant literature available. We will not discuss specific Agile methods beyond describing Agile and providing a list of the most common Agile methods. We do, however, provide some helpful hints on considerations that need to be addressed when deciding to use Agile in the DoD environment.

The audience for this paper is:

- Senior DoD acquisition policy makers, to advise them on the practicality and policy pitfalls of encouraging the application of Agile software development methods in their programs
- Members of DoD program offices who may be challenged to undertake a software development acquisition with a contractor who will be using Agile software development methods
- Software development contractors who are contemplating responding to a DoD Request for Proposal (RFP) with a proposal based on using Agile software development methods

Agile and the DoD

Agile has existed for many years, and, in fact, is based on concepts that have been around for decades. Agile achieved its greatest success in small- to mid-sized, commercial applications. There has been limited documented usage in the DoD/government arena.

In recent years, Agile matured and personnel became skilled in applying Agile; some DoD contractors started to build internal Agile capabilities and use Agile on DoD programs. Some DoD acquisition programs proposed and used Agile processes, attempting to take advantage of contractor capabilities, but without (as yet) any formal DoD guidance, templates, or best practices.

Given this backdrop, can Agile produce a better product developed within cost and schedule parameters? If barriers interfere with the DoD adopting Agile, how can they be addressed?

Our interviews and research into whether Agile can benefit the DoD resulted in a resounding, but qualified, “Yes.” Agile *is* another tool that can provide both tactical and strategic benefits. The tactical benefits of lower cost within schedule and increasing quality are important; however, the strategic benefits of being responsive and being able to adjust to the current situation more rapidly might be of even greater value. This could be a huge factor in today’s world, where the DoD needs to get results faster and be better aligned with changing needs. In fact, reports[1] available about Agile are impressive. Even if experience

[1] Several results show that by using Agile methods costs decrease from as little as 5 to as much as 61 percent, with schedule decreasing from as little as 24 to as much as 58 percent, and cumulative defects decreasing from as little as 11 to as much as 83 percent [**8**].

provides savings for DoD programs on the low-end of the spectrum, these savings can be significant over time. We also found that there are no prohibitions for using Agile in the DoD 5000 series. In fact, the IEEE[2] is currently working on a standard for Agile. To date the standard is unpublished, but the fact that the IEEE has deemed it *worthy* of a standard is a step in the direction of obtaining formal guidance for Agile.

During our research we noted that in the current, traditional Waterfall method commonly employed within the DoD, there is an established practice that uses some form of controlled experimentation. Current Waterfall practices create experimental code or prototypes and then throw them away. Rather, Agile builds software iteratively, refining or discarding portions as required to create increments of the product. The idea is to have some working code at the end of each iteration that can be deployed. There are some programs within the DoD today that are employing Agile to do just this.

Embracing Agile Methods

Agile processes are based upon good ideas derived from successful industry practices. We believe the DoD should embrace Agile for some programs and traditional Waterfall methods for others. There is no "one size fits all" Agile process. Just like any set of processes, implementation of Agile must be tailored to fit the situation and context. For example, Agile teams responsible for developing high-risk, core components of the software architecture might apply less-aggressive release schedules than Agile teams developing less critical pieces of the software system. Some Agile teams might pick a two-week iteration cycle where others might determine their optimum iteration cycle is three weeks. Agile is not a silver bullet but rather another "lead bullet" for the Program Management Office's and contractor's arsenal.

Sometimes a hybrid approach of traditional Waterfall methods and Agile is the best for a particular program. For example, due to safety considerations some mission critical systems might require certain traditional milestones and documentation deliverables to remain in place. However, Program Management Offices (PMOs) might work with the Agile development teams to agree upon a hybrid approach that bridges these requirements with the need for agility and responsiveness. Perhaps the PMO agrees upon fewer, less formal reviews and delivery of smaller sets of high-value documentation in exchange for getting a beta version the user can start evaluating in the field more quickly.

Moving to Agile will require considerable work from the DoD entity (PMO, DoD, OSD, and perhaps Congress), and is not without hurdles, most notably the following:

Acquisition Life Cycle

Each life cycle phase (e.g., Materiel Solution Analysis, Technology Development, Engineering and Manufacturing Development, Production & Deployment, and Operations & Support) presents unique challenges and opportunities. Some phases lend themselves to the use of Agile better than others. You must consider the Agile processes and practices you want to use early in the acquisition life cycle; it is of critical importance to make sure that contractually binding documents, such as RFPs and Statements of Work (SOWs), support those processes and practices. For example, if you embrace Agile you need to determine how you will meet the standard milestone criteria such as PDR and CDR. Typically, the types of documentation expected at these milestone events are not

[2] IEEE P1648 is a draft standard. See http://standards.ieee.org/announcements/pr_1490p1648.html

produced using Agile. Thus, you should create expectations and criteria that reflect the level and type of documentation that would be acceptable for those milestones and yet work within the Agile constraints.

Team Environment

A central concept to Agile is the small, dynamic, high-performing team. The challenge is this: *How do I provide an environment that fosters the creation of self-forming or dynamic teams in a culture that is accustomed to static, centralized organizational structures?* To complicate this further, consider that the software team might be a small part of an overall system procurement for something like a tank, ship, or plane. The system environment might call for centralized configuration management, highly defined legacy interfaces, and a predetermined architecture, all of which constrain the software. This environment, then, should be treated as a constraint by the Agile team and can provide boundaries within which the Agile team can operate. These system boundaries could act to encapsulate the Agile team environment.

End-User Access

Access to end users can be complex and difficult when dealing with any single service but it can be even more complex with multi-service programs. Agile developers need to have a single voice for the user and one that can commit to changes for the product being developed. In some Agile approaches, the “one voice” is a product owner or manager who brings decisions to the Agile team that have been made through collaborative interaction. Within the DoD, the acquisition organization typically is the proxy for the end-users and only duly warranted personnel can commit the program. To mitigate these issues, end-users should be invited to all demos where they can provide feedback that only becomes binding with PMO approval. The end-users need to work closely with the PMO, as with any acquisition.

Training and Coaching

While Agile concepts may not be new, the subtleties and nuances of each Agile method can be new to the uninformed PMO. To overcome this, train the PMO staff before starting and employ an experienced coach or knowledgeable advocate for the PMO to help guide them throughout the process. It is important to set aside funding for initial and ongoing training and support.

Oversight

Traditional methods have well-defined oversight methods. Agile oversight methods are less defined and require more flexible oversight to accommodate the fluidity of Agile implementation. Resolution of the specific type of oversight needs to be done in advance. One aspect of the Agile management philosophy is that the primary role of manager is more of a team advocate than overseer. The management function of roadblock remover is critical to the success of an Agile team. Thought needs to be given to what day-to-day PMO activities might need to be altered to support this type of change.

Typically, documentation is used by the PMO throughout the development cycle to monitor the progress of the contractor. Documentation produced using Agile methods is just enough to meet the need and provide continuity for the team. This type of documentation is usually not sufficient for the capstone reviews or monitoring progress. The PMO needs to create different ways to meet the same objectives for monitoring while leveraging the advantages of Agile.

Rewards and Incentives

Agile rewards and incentives are different from the typical structure of traditional methods. In the DoD environment, the challenge is finding ways to incentivize teams and individuals to support Agile goals such as innovation, rapid software delivery, and customer satisfaction. At the same time, we need to eliminate rewards that incentivize the wrong things. For example, rather than rewarding contractors for fixing defects we may want to reward the developer for early delivery of beta software to a limited set of users in a constrained environment. This way the beta users get to test the product in the field sooner while at the same time providing feedback that helps to improve the quality of that iteration of the software. One other type of incentive that should be considered is incentives that encourage a collaborative relationship between the PMO and the contractor's staff.

Team Composition

The composition of the PMO staff might look somewhat different in order to accommodate the use of Agile. The government should consider adding a knowledgeable Agile advocate or experienced coach to their team. End-user representatives are essential for Agile. This position will be difficult to fill in a timely and consistent manner. Some programs have used rotating personnel to fill this position.

Another challenge is keeping high-performing Agile teams together long enough for them to achieve peak performance. This is a challenge because developers change at the end of a contractual period of performance. One recommendation might be to look at putting key Agile technical developers or technical leads under a separate contract vehicle or hire them to work for the government organization itself.

Culture

The overall organizational culture needs to support the Agile methodology in use. The Agile culture is counter to the traditional Waterfall culture in everything from oversight and team structure to end-user interaction throughout development. This will require a mindset change for the PMO and other government entities such as OSD. In order to employ any of the Agile concepts, the DoD organization will have to plan for them, anticipate the changes needed in their environment and business model, and apply the hard work to make the changes a reality. Organizational change management is essential during the transition to Agile.

1 Overview

Agile methods for software development have existed for many years. These methods have achieved their greatest success in small- to medium-sized commercial applications. To date, based on our research, they have had limited usage and success in the DoD/government arena.

In recent years, as Agile methods have matured, personnel have become more skilled, and education/training programs have become available, some DoD contractors have begun to build internal Agile capabilities and initiate pilot usage efforts on DoD programs. Many DoD acquisition programs have also begun to propose and use Agile processes, attempting to take advantage of contractor capabilities; however, they have done this without (as yet) any formal DoD guidance, templates, or best practices.

In 2009 the Carnegie Mellon® Software Engineering Institute (SEI) was tasked by Mr. Blaise Durante, SAF/AQX, to assess the state of the practice of Agile development in government software acquisitions. This team was assembled to complete that assessment.

This report provides the results of the SEI study of the current utilization and future applicability of Agile for software development in DoD acquisitions. The study was conducted in the latter half of 2009 and completed in January 2010. This report is intended for:

- Senior DoD acquisition policy makers, to advise them on the practicality and policy pitfalls of encouraging the application of Agile software development methods in their programs
- Members of DoD program offices who may be challenged to undertake a software development acquisition with a contractor who will be using Agile software development methods
- Software development contractors who are contemplating responding to a DoD Request for Proposal (RFP) with a proposal based on using Agile software development methods

1.1 Original Tasking and Approach

Our team set out to document lessons learned and/or best practices in as many programs as we could find in the DoD acquisition community that are using or have used Agile for software development. For this report, we made the assumption that we were dealing with software only or software intensive systems. Our purpose was to answer two questions:

- Is the use of Agile beneficial to the DoD; that is, can it produce a better end product developed within cost and schedule parameters?
- If the answer is "Yes," what are the barriers to using Agile in the DoD acquisition environment, and how might these barriers be addressed?

Our approach was to address both questions simultaneously because we believed that, regardless of any of the *theoretical* benefits of Agile (and it was quickly evident that there were many), it would only remain an academic interest if there were not solid experience available on the *actual use* of Agile within the DoD acquisition environment. Thus, we looked for current and recent DoD software development

® Carnegie Mellon is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

acquisitions that claimed to have used one or more of the methodologies generally accepted as Agile (e.g., eXtreme Programming, Scrum, Lean Software Development, and others[3]). We found a small number of such programs willing to share their experiences; these were weapons systems programs including Joint Mission Planning System (JMPS), Single Integrated Air Picture (SIAP), Operationally Responsive Space (ORS), Virtual Mission Operations Center (VMOC), Space Radar, an Army tank program, and some other classified programs. The programs ranged in size from small to fairly large. The amount of detail we were able to obtain from each program was a function of program constraints such as security. With this in mind, we conducted interviews with development team members, program office personnel, or other members of the SEI staff who had intimate knowledge of those programs to do the following:

- Document lessons learned and/or best practices from the case study of the multiple programs using Agile, to include contractor capabilities, government management (strengths and weaknesses), and conflict points with standard DoD methods, etc.
- Examine the viability of developing an approach that can be used within the DoD 5000 acquisition environment to take advantage of the benefits of using Agile methodologies with minimal need for policy waivers.
- Provide guidelines on how DoD technical milestone reviews (SSR, PDR, and CDR) should be altered/augmented to account for Agile software development practices.

After our interviews, we prepared an "*Agile Study Lessons Learned*" draft presentation and sent it to the interviewees for comment. We did this to assure that we understood the points that interviewees made and to lay out an initial argument addressing the primary questions. We incorporated comments we received resulting in a final, annotated version of the presentation, which was given limited distribution. Finally, we created this report to document our findings and provide recommendations.

1.2 What Is Not Addressed

"Agile," in the context of software development, is a term that encompasses many different tools, techniques, and methods, some of which are briefly described in Section 2: What is Agile? We give the reader context and awareness of Agile, but do not attempt to provide a comprehensive review or tutorial of specific Agile methodologies for application in software acquisitions. Rather, we provide questions to ask and guidance on how Agile could be useful and have relevance to DoD organizations.

We have not attempted to address the question of whether DoD PMOs themselves could become "Agile" in their own internal operations. While this was discussed at length, we decided that such a discussion would go too far beyond the current experiences of the interviewees.

We have also not attempted to discuss the relationship between CMMI and Agile software development methodologies. In our view, CMMI is a framework of best practices that can be applied in any software development program (whether or not that program uses Agile). We recognize that some of the Agile advocates have equated CMMI with "traditional" and/or "Waterfall" software development approaches [**1**], but we believe that this is only a misunderstanding of CMMI on their part. Others have suggested that Agile and CMMI should be embraced together [**2**].

[3] Appendix A contains several examples of Agile methods.

2 What is Agile?

On the surface it seems that there is really nothing "new" about Agile. However, upon close inspection there are new components (ideas, practices, theories, etc.) and new combinations of those new components with "old" components. The explicit value statements used within Agile are also new. However in a way, Agile has simply swept up software development practices that have been used since the earliest days of software and added a few new twists. Philosophically, Agile also borrows heavily from approaches that have been successfully used in manufacturing throughout the world for decades, such as "just-in-time," Lean, Kanban, and work-flow-based planning. Another new development is that Agile is becoming codified, evolving from a collection of disjoint, separately developed software development methods into a philosophically coherent family of such methods.

This philosophical coherence—and the current energy driving advocacy of Agile—was the result of a remarkable meeting among thought leaders and consultants[4] in software development who would normally have been competitors. In February 2001 seventeen people met to try to find common ground and ultimately produced the *Agile Software Development Manifesto*. This "manifesto" detailed all of their commonalities overlooking, for the moment, areas where they had differences of opinion.

2.1 Agile Manifesto and Principles—A Brief History

The self-named Agile Alliance shared allegiance to a set of compatible values promoting organizational models based on people, collaboration, and building organizational communities compatible with their vision and principles.[5]

Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

Jim Highsmith asserts that "the Agile approaches scare corporate bureaucrats—at least those that are happy with pushing process for process' sake versus trying to do the best for the 'customer' and deliver something timely and tangible 'as promised'—because they run out of places to hide."[5]

As the Agile Alliance noted, the four dichotomies listed in the manifesto ("individuals and interactions over processes and tools") are not intended to suggest that what is on the left is important and what is on the right is unimportant; rather, what is on the right, while important, is simply less important than what is on the left. For example, some

[4] The signatories were representatives from Extreme Programming, SCRUM, DSDM, Adaptive Software Development, Crystal, Feature-Driven Development, Pragmatic Programming, and others: Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, Dave Thomas.

[5] http://agilemanifesto.org/history.html

believe that the Agile approach advocates providing no documentation other than the code itself. The Agile community would argue instead that documentation *is* important, but no more documentation should be created than is absolutely necessary to support the development itself and future sustainment activities. In fact, Agile emphasizes collaboration and the notion that when documentation replaces collaboration the results are problematic. Documentation should be the result of collaboration.

The Agile Alliance says the following principles underlie the Agile Manifesto:

> *Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.*
>
> *Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.*
>
> *Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.*
>
> *Business people and developers must work together daily throughout the project.*
>
> *Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.*
>
> *The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.*
>
> *Working software is the primary measure of progress.*
>
> *Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.*
>
> *Continuous attention to technical excellence and good design enhances agility.*
>
> *Simplicity—the art of maximizing the amount of work not done—is essential.*
>
> *The best architectures, requirements, and designs emerge from self-organizing teams.*
>
> *At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.*[6]

From these principles, it is understood that Agile is really a *philosophy* or *development approach*, and it comprises a number of more specific methods, for example, eXtreme Programming (XP), Scrum, and Adaptive Software Development (ASD). (A synopsis of Agile methods is provided in Appendix A.)

6 http://agilemanifesto.org/principles.html

2.2 A Practical Definition

While this history provides a context for Agile, it does not provide a specific definition. The authors struggled with defining Agile. Since there are plenty of definitions to choose from, we picked one that closely reflects our intended use of the term "Agile" within this paper and one that is more concise:

> *Agile: An iterative and incremental (evolutionary) approach to software development which is performed in a highly collaborative manner by self-organizing teams within an effective governance framework with "just enough" ceremony that produces high quality software in a cost effective and timely manner which meets the changing needs of its stakeholders.*[7]

This definition is rather long but it covers our purposes. If a shorter definition is desired by the reader, Alistair Cockburn has said that Agile is

> *…early delivery of business value. That involves early and regular delivery of working software, a focus on team communications, and close interaction with the users.*[8]

2.3 Example Agile Method

In order to provide the reader with further context of how Agile might be used, a simplistic generic example for a software development project might include the following.

Initial planning[9]

- The overall scope of the project is examined. The business side sets the overall priorities and the development team members select and estimate work items.
- A fixed iteration length is determined (usually between one and four weeks; a two-week iteration appears to be common).
- The functional scope is broken down into a set of "user stories" (capabilities) that initially are described in a coarse-grained manner. Prior to implementation within an iteration, the "stories" are elaborated at a level detailed enough to allow each story to be implemented within a single iteration
- The highest risk and highest priority "stories" are moved to the front of the queue for implementation in the development iterations. The "stories" at the front of the queue tend to be those with highest stakeholder value, which would include priority and risk.

Iterations[10]

- Each iteration starts with the team planning session; stories are selected from the queue until a full iteration's worth of work is identified.
- Each story (capability) is refined further into specific tasks as noted above.

---

[7] http://www.agilemodeling.com/essays/agileSoftwareDevelopment.htm

[8] http://bradapp.blogspot.com/2006/05/nutshell-definitions-of-agile.html

[9] For large programs, initial planning would be done during Iteration 0. Iteration 0 is a planning iteration only. Release planning, overall program planning, and high-level architecture creation are some of the tasks accomplished during Iteration 0.

[10] Note that the customer or user is available for feedback throughout the iteration.

- Development work first begins with the writing of unit tests that will be used once the software is developed if using the Test Driven Design [TDD] method.[11]
- Coding does not begin until the unit tests have been written (if using TDD).
- At the end of the iteration, the output is an executable, testable product that could actually be used and unfinished stories could slip into the next iteration. Typically, automated testing is used extensively in Agile.
- A retrospective is usually held at the end of the iteration. The retrospective gives the team an opportunity to reflect on the iteration and determine lessons learned (what went well and what needs to improve).

2.4 Waterfall and Agile Methods Compared

All authors and Agile advocates compare and contrast Agile techniques and methodologies with the Waterfall model of software development. We will only discuss a few of the differences.

For our purposes, the definition of the Waterfall Model (referred to as Waterfall) is

> *A software life-cycle or product life-cycle model, described by W. W. Royce in 1970, in which development is supposed to proceed linearly through the phases of requirements analysis, design, implementation, testing (validation), integration and maintenance.*[12]

The Waterfall development model[13] has its origins in the manufacturing and construction industries, highly structured physical environments in which after-the-fact changes are prohibitively costly, if not impossible. Since no formal software development methodologies existed at the time, this hardware-oriented model was simply adapted for software development.[14]

2.4.1 Frequency of Usable Releases

One of the primary differences between Waterfall and Agile is the frequency with which usable releases are produced. With Waterfall, development goes through a series of distinct phases: requirements analysis, design, implementation, testing, integration and maintenance. For instance over a two-year project duration, each phase might range from three to six calendar months. At the conclusion of each phase, a formal milestone review is typically conducted as a capstone event (also a user validation). There is only one product release at the end of testing.

[11] For more information see: http://agiledata.org/essays/tdd.html or http://en.wikipedia.org/wiki/Test-driven_development

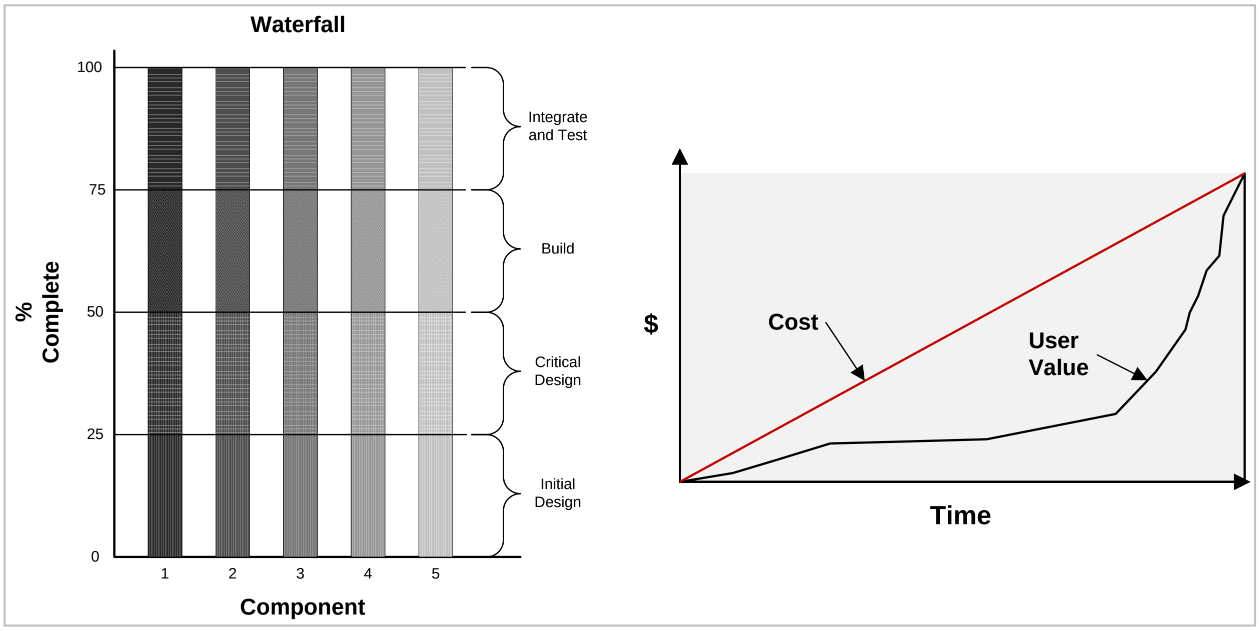
[12] http://www.websters-online-dictionary.org/Wa/Waterfall+Model.html

[13] The first formal description of the waterfall model is often cited to be an article published in 1970 by Winston W. Royce (1929–1995), although Royce did not use the term ""waterfall"" in this article. Royce was presenting this model as an example of a flawed, non-working model (Royce 1970). This is in fact the way the term has generally been used in writing about software development—as a way to criticize a commonly used software practice.

[14] http://en.wikipedia.org/wiki/Waterfall_model

In contrast, Agile breaks the development into a series of short iterations, between one and eight weeks (most typically two weeks), that produce a usable product at the end of each iteration.[15] The idea is that the stakeholders receive usable code much sooner when using Agile.

As shown in Figure 1, Waterfall typically decomposes the system into subsystems that address specific requirements in "stovepipes" and provides the user with limited user value until subsystems are integrated at the end of the entire project. All requirements established during the life cycle phases are considered to be equally "required," regardless of their ultimate value to the end user or their risk to the project if they cannot be successfully achieved. If strictly followed, Waterfall culminates in "big bang" integration, in which any issues not anticipated in the pre-coding phases emerge.



16

Figure 1: Waterfall and Resulting Value/Time Curve

2.4.2 Executable and Testable Product

Another significant difference between Agile and Waterfall is that the output of each Agile iteration is an executable, testable product that could actually be used. Therefore, the original scope of work could be modified dynamically by the project team. For example, the customer could decide that the cumulative scope of work built into an iteration might be all that is actually required, and could potentially terminate the project at the conclusion of any iteration. Conversely, scope could be added, or priorities and deliverables could be modified. Figure 2 shows a view for Agile, where a snapshot was taken after each iteration to show the actual state of the various subsystems (or components) that make up the software system.

[15] Usable means code that provides a coherent piece of functionality. However, the code from a single iteration might not have sufficient functionality to be useful to external stakeholders. Sufficient functionality might only be available through releases (multiple iterations) which are provided to external stakeholders.

[16] The value/time graph was adapted from Jim Highsmith.

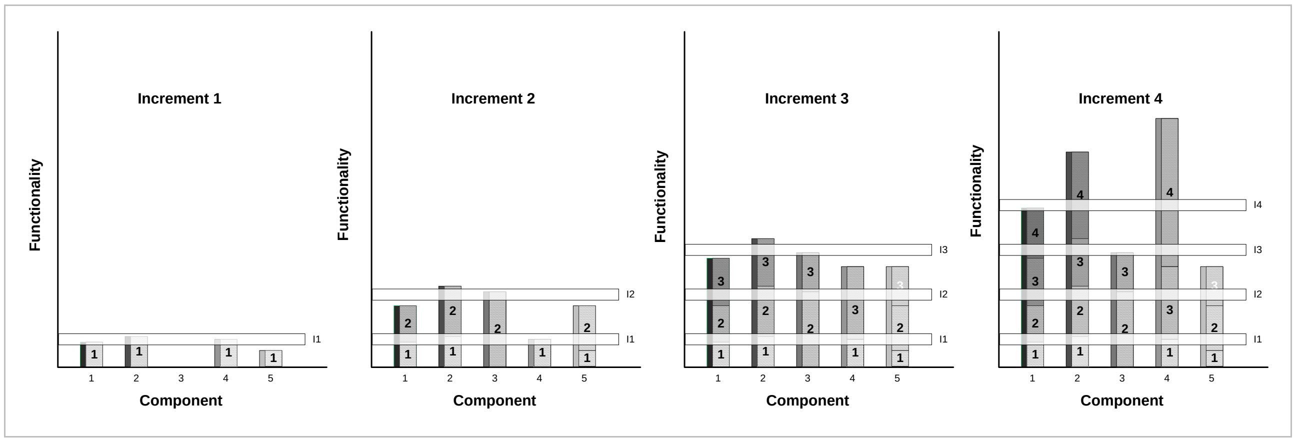


Figure 2: Agile/Incremental Build

Because Agile forces an executable product to be produced, the Agile team learns about integration and testing issues very early in the project. Lessons learned from early integration and testing will influence future iterations.

If we were to extend the iteration builds to the same scope as we included in the Waterfall diagram shown in Figure 1, we arrive at the depiction in Figure 3. The "stories" that typically encapsulate the system requirements in Agile cut across the various elements of the business logic structure and incrementally realize user value. Optimally, high-value and high-risk requirements have been completed as part of stories completed early in the project, and system integration has been carried out at the end of each iteration, risk has been steadily worked off (at least in principle), and the user value of the project has been achieved incrementally as the project progresses.

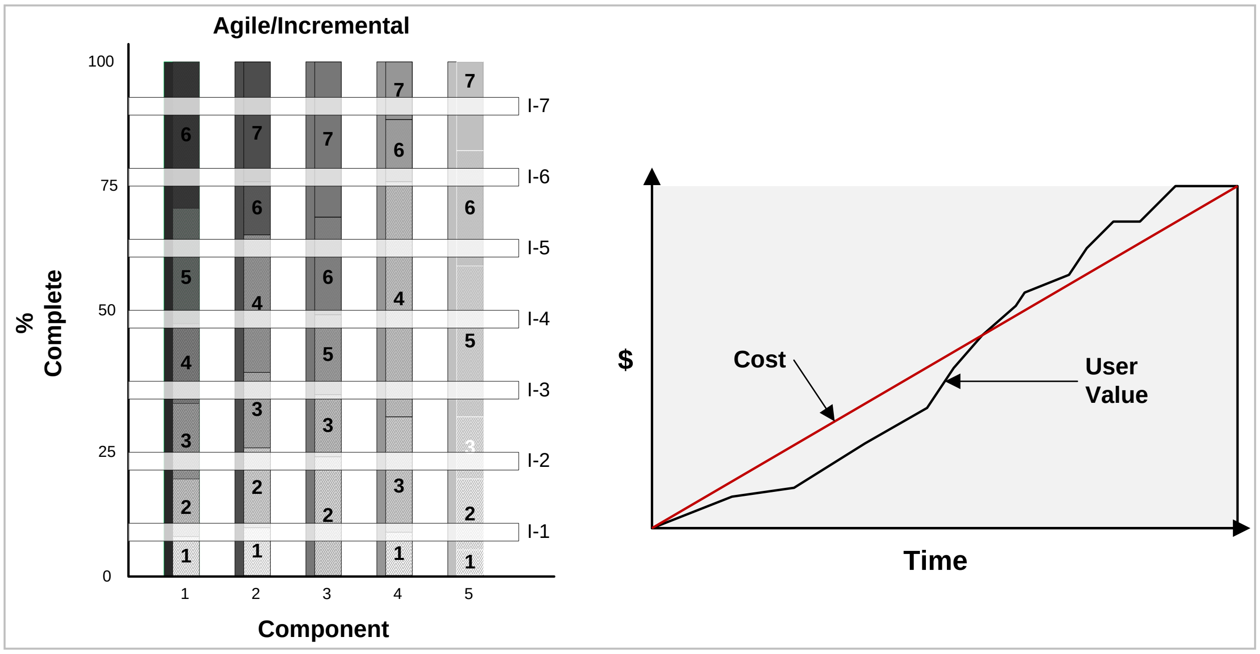


Figure 3: Agile and Resulting Value/Time Curve[17]

[17] The value/time graph depicted in this figure was adapted from Jim Highsmith.

2.4.3 Internal Integration and Test Group

One additional point can be made to illustrate the difference between Waterfall and Agile. If we were to compare two project teams working on equivalent scope (one using Waterfall, and the other using Agile), we should expect their staffing profiles to be quite different. Because each Agile iteration produces an executable, testable product, there is obviously more staff members that are usually allocated to the integration and test functions for both internal components and components of the system early in the life cycle. The Agile workforce seems to have a more stable distribution of effort among the various disciplines than when using Waterfall. Figure 4 provides a notional illustration of how this might appear for the Integration and Test groups.

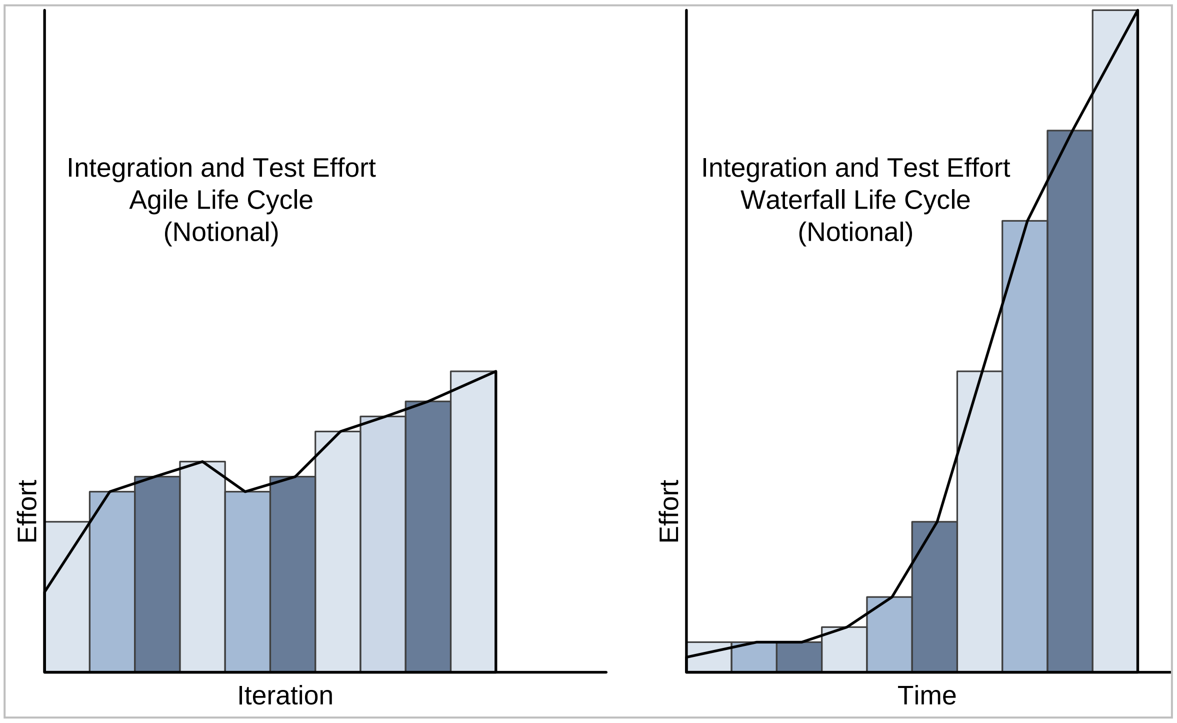


Figure 4: Notional Illustration of Agile Versus Waterfall Integration/Test Efforts

To a certain extent, the difference is that, with Agile, the integration and test are largely internal to the Agile team, while with Waterfall integration and test is a separate group. The reality is that when you begin integration and test activities, you begin to identify unintended behavior (defects) that require effort to fix. In Agile, the effort associated with this rework takes away from the time available to build new functionality. Under some conditions, this effect could escalate and lead to iterations that do nothing but fix defects.[18] On the other hand, in Waterfall, waiting until the end of the life cycle to perform the integration and test activities can lead to extensive delays when there are significant problems found; this can result in significant rework. Obviously, arguments can be made that support either side in the analysis of this distinction between Waterfall and Agile. The rationale behind Waterfall is that with more time to anticipate integration issues, a better design could have been taken from the outset, avoiding the defects entirely. The rationale behind the continuous integration approach is that a certain percentage of defects would

[18] The team uncovers the issues but the users/customer/business side decides what is done to address the issues. They could decide to fix them and reduce or eliminate functionality to allow time for the fixes.

not have been created or would have been found sooner since integration and test start sooner. The early detection of these defects results in significantly less overall effort to resolve them.

As we have discussed, Agile can have multiple advantages over using Waterfall. So, then, why are so few DoD software acquisitions using Agile? For those DoD software acquisitions that have used Agile methods, what lessons have they learned? We interviewed a select group of programs about their use of Agile, and what we learned is grouped into the following topics, which are discussed in more detail in the next section.

- Acquisition
- Knowledge of Agile
- Culture
- Oversight
- End-User Involvement
- Integration and Test
- Infrastructure

3 Interview Observations and Findings

The results from interviews with Agile practitioners and other observations of Agile within the DoD environment are presented in this section. The results are lessons learned from actual application of Agile within the DoD environment. We reviewed the interview notes and other observations and found seven common topics, which we use to frame the discussion of the overall results. Each topic is structured to provide a context and the associated finding/observation. These topics are related to each other and there is overlap between their findings.

Note: We also provide additional information about Agile concepts and how to apply them in Section 5; Section 3 **only** presents what we learned from the interviews and observations.

The material in this section was derived from interviews that we conducted with the following people and organizations:

- BAE—experiences as contractor for Joint Mission Planning System (JMPS) and other programs using Agile methods
- JMPS Program Management Office (PMO)—experiences with Agile on JMPS
- Fred Schenker—SEI technical staff experiences with Single Integrated Air Picture (SIAP) program
- Harry Levinson—SEI technical staff experiences with Joint Mission Planning System (JMPS) Independent Technical Assessment
- Mike Bandor—SEI technical staff experiences with Operationally Responsive Space (ORS) , VMOC (Virtual Mission Operations Center), and Space Radar programs
- Keith Korsac—SEI technical staff experience with army tank program
- Major Daniel Ward (now Lt Col)—discussions of FIST applications

3.1 Overview of Common Topics

Acquisition

It is widely believed, both by program offices and DoD contractors, that the DoD 5000 series and other regulations and guidance documents limit the government and contractors from using a non-Waterfall approach. A particular sticking-point is that Agile does not readily accommodate large capstone events such as Critical Design Review (CDR). However, the programs that have used Agile in software development have found that the DoD 5000 series has great flexibility and does not, in fact, preclude the use of Agile.

Knowledge of Agile

Agile methods have been developed and used most extensively in the non-DoD commercial sector with small- to medium-sized projects. Experience with larger projects has recently started to accumulate in the commercial sector. As a result, few DoD acquisition professionals and DoD contractors are familiar with the use of Agile or possess the necessary experience to effectively implement it.

Culture

Culture is the customary knowledge, beliefs, behaviors, and traits displayed by an acquisition organization or contractor. The government is heavily invested in the use of Waterfall for acquisition of all equipment and systems, whether they are software intensive or not. As a result, a large segment of the DoD acquisition community (and that of long-time DoD contractors, as well) is more comfortable with Waterfall and skeptical about using Agile.

Oversight

Tracking and measuring progress while using Agile in a way that is clear to and trusted by the government is a particular challenge. The metrics applied on past software acquisitions, including the use of the Earned Value Method System (EVMS), do not work well for Agile at best, and at worst do not work at all. Agile also does not support the kind of granularity of estimates and task detail that is typically shown across the entire project in an Integrated Master Schedule (IMS). Rather, Agile provides high granularity task-level estimates for just the upcoming iteration.

End-User Involvement

The close involvement of end-user participation in the development process, reviews, and demonstrations, upon which successful Agile implementations depend, is extremely difficult to achieve with the many stakeholders typical of DoD acquisitions. In addition, the continuous availability of the end user is an issue in the DoD environment, as end users are usually in operational not acquisition organizations. Acquisition personnel tend to be isolated in acquisition organizations, rather than integrated into operational units.

Integration and Test

Because integration and testing activities are part of Agile development iterations, the approach to these activities might significantly change from those used in Waterfall. The biggest change is that integration and test need to be done throughout the project as opposed to waiting until the end of the release cycle. This is another particularly challenging issue of culture-change since, historically, integration and test organizations have been outside the development teams once you get beyond unit testing.

Infrastructure

No matter whether one uses Waterfall or Agile is used, the group undertaking the project needs to have an infrastructure supporting it. This would include the organization of the team and the context within which the team operates. The overall organization of a project using Agile is different from the traditional program structure. The structure for an Agile project reflects Agile precepts and is reliant on the context in which it will be applied.

3.2 Acquisition

Context

A strong belief that is prevalent across the DoD community is that the DoD 5000 series, and other acquisition policies, instructions, and regulations are rigid in requiring a traditional Waterfall process for the development of software. It is true that most acquisition personnel have been specifically trained in applying the Waterfall method to these acquisition regulations, irrespective of whether they are acquiring tanks and aircraft or acquiring software that might be used in stand-alone software applications, such as pure IT acquisitions (e.g., enterprise resource planning or personnel/pay systems).

Finding/Observation

Those programs that have used Agile in software development have found that the DoD 5000 series has great flexibility and does not in fact preclude the use of Agile. It appears that with careful review and some tailoring an alternate interpretation can be created so that Agile can be used on DoD programs.

Context

An interesting corollary to the prevalent belief of using traditional methods is that many Request for Proposals (RFPs) are written in such a way that a non-Waterfall response would appear to be or might be noncompliant. Most traditional RFPs require a full complement of Contract Data Requirements Lists (CDRLs) for documentation of progress.

Finding/Observation

This level of documentation is contrary to Agile precepts of creating “just enough” documentation. “Just enough” will vary from situation to situation depending on the needs and regulation requirements of the project. In order for Agile to become common place within the DoD, the acquisition organizations should encourage, and contractors should provide, a compliant proposal with suggested alternatives that use Agile. It is important for the acquirer to understand Agile benefits and to include project-specific guidelines[19] in RFP language for how Agile responses should be framed.

Context

A very specific acquisition issue and sticking point is that Agile methodology does not accommodate large capstone events such as Critical Design Review (CDR), which is usually a major, multi-day event with many smaller technical meetings leading up to it. This approach requires a great deal of documentation and many technical reviews by the contractor.

Finding/Observation

A software developer using Agile typically does not complete the design before beginning implementation of it, so the scale and comprehensiveness of a CDR is quite foreign to Agile development teams. Some experienced Agile providers have accommodated this issue by breaking the typical Waterfall-based CDRs into multiple Interim Design Reviews (IDRs), which is an example of the type of “flexibility” in implementing DoD 5000 requirements. These IDRs need to reflect the iterative nature of Agile, and they can be held more frequently and with tighter focus for only a few hours at a time, as opposed to the several days needed for CDRs. The entry and exit criteria for an IDR need to be dependent on the current iteration, and the results of a combination of all IDRs completed should be functionally equivalent to a CDR using Waterfall.

[19] As of publication, the authors are not aware of the existence of any guidance for how to frame Agile responses.

3.3 Knowledge of Agile

Context

A good understanding of the fundamentals of Agile development methods is required by both the government and contractor personnel. Without this understanding and knowledge, misunderstandings will certainly happen and could have disastrous consequences.

Finding/Observation

One example we found of a typical misunderstanding arose as early as contract negotiations. The contractor specified that they would deliver documentation in accordance with best Agile practices. The government included a list of required documents in the contract. The contractor understood that the Agile best practice meant that there would be minimal detail and documentation (what, when, and how much detail would be done) up front. Even though the contractor understood this, the government did not and still expected all the detail that was traditionally provided for the documents. The government had trouble accepting less detail and documentation even though the complete content would become available at a later date.

The issue was not only how many documents were produced but the expected level of detail in the documents and when the documents would be complete. One way to increase the amount of knowledge of Agile and avoid this type of misunderstanding is through education. For example, courses at the Defense Acquisition University (DAU) and other institutions could be updated to include discussions of Agile, its pros and cons, and the challenges Agile presents to the Program Management Office (PMO). One such pro is that the Agile forces closure on requirements analysis for the iteration and flushes out problems early in each iteration; these are very desirable attributes. One potential con is the resistance to the amount of culture change that may be required to employ Agile.

Context

The acquisition community's imperfect understanding of Agile might undermine the success of an Agile contractor. Agile is relatively new and has its genesis in the software development community itself, which is mostly isolated from acquisition concerns. As a result, relatively few acquisition professionals have direct experience with Agile, and such Agile-unaware PMO members might insist on the more familiar project plans and metrics, but they will not fit into Agile.

Finding/Observation

All government and contractor personnel need to spend the time necessary before contract award to understand what it means to use Agile from all perspectives. The following are examples of such considerations.

- Which contractual phases can employ Agile?
- What are the milestone and deliverable details for each phase when using Agile?
- What contract changes would be needed?
- What changes to the approach of monitoring development progress will be needed?
- What type of staff members are needed on both sides (government and contractor)?
- Which of the 5000.02 process formalities will be tailored?

Context

Both government and contractor personnel need to acquire an appropriate skill set to support Agile use within DoD systems. The nature of Agile lends itself to a slightly different staffing model than the one the government is used to seeing with Waterfall.

Finding/Observation

From our interviews, we learned that there are subtle but critical differences:

- The contractor's program manager needs to be experienced in Agile. The government program manager should also be experienced in Agile, though at present this might be more than can reasonably be expected.
- Contractor personnel need to be trained and experienced in the Agile method to be used on the project.
- PMO personnel need specific training in the Agile method that the contractor is going to use, as well as a more general understanding of Agile. They need to develop an understanding that Agile is adaptive to each project or program.
- There needs to be an expert advisor/advocate for Agile in a position of authority in the PMO. Without authority, such an advisor/advocate becomes "just another opinion."

3.4 Culture

Context

The government is heavily invested in the use of Waterfall for acquisition in general, and this has been applied to software acquisition as well. While other methods have been used, Waterfall and its accompanying precepts are most familiar to most PMOs.

Finding/Observation

Moving to Agile is difficult—many of the "old ways" and paradigms need to be modified using a fundamental culture change. As alluded to previously, the existing training in the interpretation of software acquisition requirements is skewed toward the Waterfall approach. Thus, a PMO employing Agile will need to be trained in Agile concepts.

Context

One challenge regarding benefits for the DoD is that the acquisition community might not perceive that there is any benefit in using Agile. Many believe Agile is *ad hoc* and that it does not produce necessary documentation or apply any rigor to development.

Finding/Observation

The PMO specifically needs to realize that while Agile provides many benefits, many of the traditional Waterfall activities, documents, etc. will not be present. In some cases, the data will be present but not in the anticipated form.

Context

Since the type of management oversight is different for Agile than Waterfall, members of the PMO are likely to feel that they are losing control over the program.

Finding/Observation

Historically, the PMO’s role is to ensure orderly development progress, but with Agile the PMO has to relinquish control over how change is managed. Agile attacks high-value and/or high-risk user items first instead of making steady progress on all requirements. This difference in handling requirements can create unnecessary friction between the PMO and the contractor, leading to outright hostility.

Context

Both the PMO and the contractor need to be aware that different skills sets or skill mixes will possibly be needed in programs using Agile (as opposed to programs using Waterfall).

Finding/Observation

Agile takes a lot of strong, focused team and management oversight at the mid- and low-levels versus the high-level, particularly if a new development project/program is using a merger of Waterfall and Agile. Furthermore, the reference estimates that PMO members have developed over time about the number of developers needed based on the size/volume of the code may not be valid in an Agile environment. The management oversight required in the developers facility is more at a technical level than at the project/program management level when using Agile—what is needed are “iteration leads,” “scrum leaders,” etc. This different skill mix does not necessarily lead to a more costly management structure, but it does require a different “scorecard” to evaluate progress and troubleshoot development issues [**3**] [**4**].

The important point here is that the PMO must be prepared to deal with organizational change management issues.

3.5 Oversight

Context

Traditional Waterfall provides significant oversight and insight into the implementation details of the program; this method is very structured so that it provides predictability, stability, and high assurance [**4**]. The execution of Agile is distinctly different from what the PMO has seen in the past on programs using Waterfall. The control and discipline comes from the Agile team itself rather than from control external to the team, that is project and higher management. As a result, the PMO will see a different way that the development is controlled, executed, and viewed.

Finding/Observation

We learned from our interviews that the PMO has to be prepared to relinquish some level of control and oversight of the program to allow Agile to operate effectively. What is needed is a system of program metrics that allows the PMO to have insight into the developer’s priorities and the development progress being made on a day-to-day basis, and this will allow the PMO to achieve an optimal balance between insight and oversight.

Context

Forecasting the project schedule when using Agile requires an entirely different approach than when forecasting the project schedule during Waterfall. Agile depends on being able to determine the content of iterations on a just-in-time basis, to use very short iterations, and to respond quickly to customers' changing needs. The creation of a traditional detailed Integrated Master Schedule (IMS) with the content of each iteration for the entire project is not done with Agile.[20] Agile does not support the kind of granularity of estimates and detail that are typically shown in a traditional IMS for the entire project. Traditional IMS estimates and the corresponding constituent tasks are very detailed and require a great deal of effort to change or to update. This is counter to the "just-in-time" philosophy used in Agile.

Finding/Observation

Our interviews indicated that the IMS can be maintained at a level that is compatible and appropriate for Agile. This may be more difficult than it appears because it requires a different perspective about when and to what level of detail the IMS should be developed.

Context

An additional impact is that the estimates at the iteration level in Agile are done by the iteration team, not just by management (team leads and higher level management)[21] as is the case in most Waterfall developments. Depending on the skill level and the amount of learning achieved within the development team from previous iterations, the estimates they produce might be much more coarse-grained than expected by the PMO.

Finding/Observation

If the project is not adopting Agile outright, then some compromise between the PMO expectation of a detailed IMS and the contractor's Agile management techniques will be needed to define a model that uses the best practices from both Agile and Waterfall. Some things that have been done on existing programs and some options the PMO could consider include:

- Traditional progress measures such as earned value and percent complete might be possible to use. Because a detailed IMS is not realistic for Agile, these measures would need to be computed differently than for Waterfall.
- Progress could be measured by the number of stories completed, though for this to be useful, the PMO would need to understand the full inventory of stories that the contractor projects for the development and be convinced that the sum of all the stories fully comprehends the project requirements.
- Progress could be measured by the accumulation of "user value" during development. Since the development team itself typically assigns the value of the stories that are completed, this might not satisfy the PMO unless they had fully concurred with the contractor-assigned story (capability) values.

[20] Agile creates detailed schedules for the current iteration. Agile does not create detailed schedules for all iterations apriori.

[21] Note that there is a difference in how estimations can be done at the iteration, release, and enterprise levels. At the iteration level, the team should always be involved. However, as the project gets bigger, the need for release- and eventually enterprise-level estimates may look more like those seen in Waterfall.

- In many Agile developments, the contractors use Agile tracking tools to keep track of progress. On one of the projects interviewed, the PMO did use those same tools in lieu of expecting paper progress reports and acquiring a progress-tracking tool of their own. This has two advantages:
  - the PMO learns and uses the contractor tools to follow progress and review designs, which reduces the work and cost for the PMO
  - the contractor realizes cost savings because he does not have to do any translation of what he sees in his own tools to what the PMO expects
- The contractor and the PMO need to negotiate common ground in order to define the needed hybrid model for the measurement system.
- The project progress measurement system to be used must be negotiated and agreed upon early in the project/program.

Context

Another form of oversight used on traditional programs is the production and review of documentation on a regular basis. At first look, Agile documentation might not meet DoD expectations and the perceived need for acquisition office oversight. Most PMO personnel expect a full complement of CDRLs, provided at regular, defined intervals or milestones using traditional methods.

Finding/Observation

A developer using Agile only creates the minimum documentation necessary to accomplish the tasks at hand, and the documentation evolves over time into a final product. Thus if Agile is to be employed, the government PMO needs to agree to less-than-full-blown documentation, as this saves time and avoids abandoning expensive documentation later. Further, the government PMO needs to relax traditional CDRL-level documentation at milestone events. Still, the parties need to negotiate documentation to ensure that important data represented in a minimal required set of documents (programmatic and technical) is gathered. This requires more software expertise of the PMO staff, who need to recognize that documentation versus functionality is a “zero-sum game”: if more documents are required, then less functionality will be delivered in the final system.[22]

Eliminating these documents and the related oversight is easy to say but requires trust that the contractor is doing it right; this requires some other mechanism that ensures the proper oversight such as the government being on-site, frequent code reviews, and frequent process checks. The Agilist might argue that the iteration builds provide the visibility needed for government oversight. But until there is more government experience with Agile methods, it will be difficult for the PMOs to relinquish the current technical documents needed for oversight.

3.6 End-User Involvement

Context

One of the fundamental principles cited in the Agile Manifesto is *customer collaboration*. In other words, Agile believes close interaction between the developers and end users is important. A basic Agile prin-

[22] This should not be construed to think that doing no documentation is an option.

ciple is “business people (users) and developers work together daily,”[23] but in the DoD acquisition environment this is rarely easy, and it may sometimes not even be possible. DoD acquisitions—especially joint service acquisition—involve many stakeholders with inherently conflicting needs. It is hard to get a single viewpoint from the *customer* because no one person truly represents the users. Plus, it is hard to get all the stakeholders (maintenance and sustainment personnel) involved in development decisions.

Finding/Observation

From interviews with existing programs we found:

- A single voice for the user/customer is essential. This could be accomplished through an input-filtering steering committee that documents decisions, insists that the user community speaks with one voice to the Agile developer (through requirements definition), and receives input from and gives direction to a single person representing the Agile developer.
- True users (not just PMO representatives) *must* attend demonstrations that are given specifically to get user information and feedback.
- A hybrid approach (something between “pure Agile” and DoD 5000 traditional methods) is needed for large systems to assure that agreements with multiple users are documented, external interfaces are documented and agreed to, and multiple contractual and programmatic constraints are honored.
- A strong emphasis on government/user participation in reviews and demonstrations is essential. These reviews and demonstrations will be of shorter duration and have a tighter focus with Agile, and this will result in more frequent reviews that reflect the nature of Agile development. For example, having eight two-hour reviews spread over time, as opposed to a single two-day event to cover the same material, would be the implementation of multiple IDRs instead of one CDR.

3.7 Integration and Test

Context

Test and integration are incorporated throughout the iteration life cycle used within Agile as opposed to Waterfall, which puts it at the end. Testing can have a significantly different role in the project depending on which Agile method is used. The big advantage in Agile testing and integration is that testing can, and usually is, started earlier because of the short timeframes for iterations; this flushes out problems more quickly. Furthermore, gathering customer feedback during the development phase (in each iteration) provides an early look at the code capabilities and helps reduce risk at the time of system integration.

Finding/Observation

We learned from our interviews that:

- Within the DoD, “sell-off” is the process used by the contractor to obtain formal acceptance of the developed product from the government, thus the government takes ownership as the contractor “sells off.” The nature of “system sell-off” from the contractor’s perspective is still the same as in Waterfall; there might be fewer sell off risks because of the frequent interactions among all parties during the demonstrations.

[23] http://agilemanifesto.org/principles.html

- Software builds are completed much earlier with Agile since each iteration produces a usable build; because of this, more frequent test and integration work can be done.
- The government test community can (and should) be involved early.
- A lesson specific to integration is that the software integrators need to have access to the ultimate target environment. This reduces issues for the development teams when they get to system-level integration. (The degree to which this is an advantage depends on the target environment and the number of platforms that are involved.)
- Access to the developers for the testing/integration team can be an issue because of the short (typically two-week) development cycles; this puts intense time pressure on the development team and should be addressed during forward iteration planning as iteration cycles are completed. Further, this might suggest that the testing/integration team members need to be part of the development team.
- Government testing personnel need to understand the differences inherent in Agile versus Waterfall to adequately adapt staff and time requirements for testing when using Agile. The government testing personnel need to be engaged at the development iteration level; they should not wait until the entire system is completed to initiate their testing work.

3.8 Infrastructure

Infrastructure is the basic framework or structure of the team and the context within which the team operates. The overall structure of a program using Agile is usually different from the traditional program structure. The structure reflects Agile precepts and is reliant on the context in which it will be applied. Organizational structure and environmental/support structure both need to be established to support an Agile implementation. The context of a program and its inherent organizational structure are related.

Context

For large programs there is the need for early decisions about the support structure including shared assets. To help eliminate configuration management issues, the government usually dictates the shared assets of models across contracts and contractors on a large program. Common facilities (or shared assets), such as common logging (for example automated logging of test messages), agreements on units of measure, data models, etc., will be used on all segments and components of large programs. Early decisions on such aspects can appear to be in direct contradiction to the tenets of Agile.[24] However, developers using Agile need to be aware of the larger, system-wide constructs during their iteration planning. Developers need to use these as inputs so that they can be accommodated during the Agile implementation. Developers also need to understand that for DoD-type programs, the Concept of Operations (CONOPS) developed for the overall system by the government strongly influences and provides the context for the capability stories used in Agile development. Therefore, an up-front operational architecture needs to be defined as part of the CONOPS, and all developers of the overall system need to understand that conventional “use cases” and Agile “stories” are different in construction and application. Agile “stories” are less-formal constructs written as informal English descriptions. “Use cases” are formal constructs including preconditions, post conditions, and detailed interaction diagrams.

[24] This may be true for small-to-medium sized Agile projects. However, for large projects Iteration 0 would be employed to work activities such as architecture.

Finding/Observation

Interviewees had several potential organizational ideas for executing Agile in the DoD acquisition environment:

- Agile can be experimented with early in the acquisition life cycle to try out what works and what doesn’t. Possible places to experiment might be during analysis of alternatives, risk reduction activities, activities leading up to Milestone B, phases in which only coding is being produced, and Advanced Concept Technology Demonstrations (ACTDs).
- For programs just getting involved with Agile, one organizational structure that worked well involved customers on-site at the contractor's facilities using a two-week rotational schedule. The interviewees indicated that such a short rotational schedule benefits the contractor team because it typically provides much better access to real users, and it benefits the PMO team because they have better insight into what is going on in development.
- To help get started with Agile, the contractors brought in an Agile expert who would be embedded in the team and then train his/her way out of the job. This way the Agile team learns by doing, not just from classroom training or books.
- The contractors created something concrete that behaves in a representative manner, such as an early version or prototype.

Large programs with multiple Agile teams had several more ideas:

- To coordinate project dependencies across multiple Agile development teams, the leaders of the development teams, who typically maintain control of the team through a daily “scrum,” can themselves become members of a team of consisting of all the team leaders (a “scrum of scrums”).[25]
- To maintain subject matter expertise and foster the cross-training of staff, team leads should be permanent, rotating the staff underneath them. This allows the cross-training of staff in all areas and maintains the team lead as subject matter experts—a “best-of-both worlds” approach.
- Planning for iterations was difficult with multiple Agile teams running in parallel and working on the same source tree. It was difficult to track feature predecessors. For example, if story A is needed before story B can be implemented, and story A was scheduled to be completed during the last iteration by another team, the development team needs to know if story A actually made it into that iteration before scheduling story B. This problem is made more difficult if both A and B are scheduled for the same integration because each team can decide for itself which stories get bumped from a particular iteration. In order to preclude this type of behavior, one group made this particular topic part of their daily team lead standups.
- Dependencies across multiple Agile teams working on a common source tree need frequent coordination. The interviewees pointed out that one possible way of doing this would be to have the Agile team leader scrum (the “scrum of scrums”) meet in a daily standup to track the interdependencies.

[25] When using Scrum techniques.

Context

Another issue that needs to be considered with multiple teams is code refactoring. “Code refactoring is the process of changing a computer program’s internal structure without modifying its external functional behavior or existing functionality, in order to improve internal quality attributes of the software.”[26]

Finding/Observation

We found that the Agile refactoring step can have unforeseen side effects: When refactoring of code is carried out as a standard step in Agile and the code involved is only part of a larger, interrelated software system, team members without comprehensive knowledge of all the interrelationships can inject serious defects that are not apparent to the development team and will not emerge until system integration. From our interviews we found that

- The Agile team needs to know at all times where their code fits with other teams’ code, and what they are affecting across the entire system configuration, including the Work Breakdown Structure (WBS).
- The use of design patterns and team training in the overall architecture of the system can improve the learning curve for new team members and can help to constrain refactoring variability.
- Multiple teams working at the same time without knowing the overall program context should be cautious when refactoring solutions that simplify the code being developed. The refactoring can seem to provide an early value for code maintainability and modifiability, but ultimately not be scalable to the larger overall infrastructure.

On large, complex systems there may be a need for an entire iteration that is devoted to refactoring after integration of the various teams' code.

[26] Wikipedia - http://en.wikipedia.org/wiki/Refactoring

4 DoD 5000 Series and Agile—Potential Issues and Conflicts

Several policies, instructions, and regulations were reviewed to determine how they might impact the use of Agile on DoD programs. In particular, the DoD 5000 series was reviewed in depth. In all cited cases, the authors tried to determine if there could be an interpretation that might preclude or limit the use of Agile. In some instances, it appeared that the policy or regulation actually encouraged the use of Agile or at the very least some of the Agile concepts. The Department of Defense Directive (DoDD) 5000.01 provides supportive, challenging, and constraining policies that would need to be interpreted and applied to the specific program. Excerpts and considerations are provided in Table 1 for the areas discussed.

4.1 Use of Agile Is Not Prohibited by DoDD 5000.01

Support

Flexibility, responsiveness, innovation, and collaboration are all terms that one might see when discussing Agile. These terms are in fact section headings in DoDD 5000.01. One could interpret these sections as encouraging the use of methods such as Agile. Other sections on Integrated Test and Evaluation, Professional Workforce, and System Engineering also support the use of Agile; at least they seem to be open to methods other than traditional Waterfall.

Challenges

There are other areas within the directive that provide challenges for the use of Agile. These are cost, affordability, and cost realism. In these areas, the policy requires the program to determine the total cost of ownership which seems to be based on knowing all requirements at a detailed level up front. Agile does not necessarily support this concept well because all of the requirements are not known at a detailed level up front. However, cost as an independent variable is an inherent part of Agile, which starts out with a high-level estimate that can be, and is, refined as the program progresses. Agile allows the developers to provide an incremental total cost estimate at a detailed level as the iterations are performed. The big challenge with moving to an incremental costing approach is that the contracting cycle takes too long for just one development iteration. So, the options are either to develop a more streamlined competitive bidding process that takes months instead of years to execute or estimate several Agile iterations based on a sample set of requirements. This requires a common understanding that the actual requirements delivered will vary depending on how the PMO and end user prioritize requirements.

Another challenging area is the Program Stability policy, which details when the Milestone Decision Authority (MDA) determines to fully fund an acquisition program; generally, this is when a system concept and design have been selected. Some might say that since the design within Agile is evolving throughout the program it therefore does not support this policy. However, Agile does provide for an overall architectural framework (sometimes in Iteration 0) so that this policy can be met using Agile. The PMO would need to work closely with the MDA on meeting this objective.

Constraints

There are other areas within the directive that provide constraints for using Agile on a program. These include independent operational test, information assurance, information superiority and interoperability. These areas address the overall "environment" or context within which an Agile development project would need to operate. These policies and their implementations for the program are constraints within an Agile development effort.

The following table provides considerations for DoDD 5000.01 policies that the PMO should investigate before adopting Agile.

Table 1: Agile Considerations for DoDD 5000.01 Guidance

DoDD 5000.01 Guidance Excerpts	Considerations
4.3.1 Flexibility “There is no one best way to structure an acquisition program…MDAs and PMs shall tailor program strategies and oversight, including documentation of program information,…to fit the particular conditions of that program, consistent with applicable laws and regulations and the time sensitivity of the capability need“	**Support** This policy provides the foundation from which a program could adapt oversight suitable for Agile development. It also provides the high-level guidance for tailoring documentation such as CDRLs which would be critical when using Agile methods.
Section 4.3.2 Responsiveness “Advanced technology shall be integrated into producible systems and deployed in the shortest time practicable. Approved time-phased capability needs matched with available technology and resources enable evolutionary acquisition strategies…. Incremental development is the preferred process for executing such strategies.“	**Support** Using Agile might allow for deployment in the shortest time practicable. This policy certainly lends itself to agile methods and is applicable in a software venue. The PMO would need to interpret it for their given program and apply appropriately.
Section 4.3.3 Innovation “MDAs and PMs shall examine and, as appropriate, adopt innovative practices (including best commercial practices and electronic business solutions) that reduce cycle time and cost, and encourage teamwork.“	**Support** This policy also seems to be an invitation to use Agile. The PMO would need to interpret it for the given program and apply appropriately.
Enclosure 1 Additional Policy, Section E.1.1.2 Collaboration “The DoD acquisition, capability needs and financial communities, and operational users shall maintain continuous and effective communications with each other by using Integrated Product Teams (IPTs).“	**Support** In the Agile environment, the iteration teams are cross functional teams consisting of programmers, testers, and others as needed. Continuous and effective communication is one of the cornerstones for Agile. This policy supports the use of Agile.
Enclosure 1 Additional Policy, Section E.1.1.4 Cost and Affordability “The DoD Components shall plan programs based on realistic projections of the dollars and manpower likely to be available in future years.“	**Challenge** This section deals with the reality of fiscal constraints and the notion that cost should be viewed as an independent variable. The MDA needs to address total costs of ownership and the user should address affordability in establishing capability needs. These items might need to be obtained in a different manner when using Agile as Agile does not do a detailed determination of requirements nor identify all requirements in “concrete” at the beginning of the project. Rather Agile refines the high level requirements defined in the beginning of the project throughout the life cycle. Thus, the costs are constantly being refined too. This policy is an issue needing attention by the PMO if Agile is to be used.
Enclosure 1 Additional Policy, Section E.1.1.5 Cost Realism “Contractors shall be encouraged to submit cost proposals that are realistic for the work to be performed…….Proposals shall be evaluated for cost realism in accordance with the Federal Acquisition Regulation.“	**Challenge** The PMO would need to be convinced that the contractor is submitted realistic costs for an Agile project given that the basis for estimating in Agile is different from the usual basis for Waterfall. Agilists tend to think in terms of fixed cost and floating requirements, concepts that are counter to traditional PMO thinking.
Enclosure 1 Additional Policy, Section E.1.1.8 Independent Operational Test Agency (OTA) “Each military department shall establish an independent operational test agency… “to plan and conduct operational tests, report results, and provide evaluations of effectiveness and suitability.“	**Challenge** The OTA needs to coordinate with the Agile team. The normal mode of doing business for the OTA will most likely have to change to accommodate Agile and the timing of available deliverable code. This should be coordinated in advance of the program start if at all possible since the test (OTA) personnel need to be part of the Agile team or at least an interfacing team. This would impact acceptance testing (scheduling of it, etc.)

Enclosure 1 Additional Policy, Section E.1.1.9, E1.1.10, and E1.1.13 Information Assurance, Information Superiority, and Interoperability, respectively.	**Constraint** These sections do not directly affect the use of Agile but do provide some constraints that need to be considered for the "bigger picture" or architecture of the entire program. Agile tends to develop small, focused pieces of functionality. However, these smaller pieces will need to fit into a bigger picture or architecture for the program, which will have outside constraints or overarching requirements that need to be met. The Agile teams need to consider how every Agile iteration fits within the bigger scheme of the program in order to avoid rework. One way to solve this problem is to use a hybrid approach that couples ideas from Agile and the traditional Waterfall to provide coverage for these areas. Another possible approach could be to include an IA expert as part of the Agile team. This may or may not be a full-time position but the expert will be needed certainly on a regular and consistent basis. Another possibility is to make sure that these requirements (non-functional) are emphasized in developing and prioritizing the backlog list.
Enclosure 1 Additional Policy, Section E.1.1.11 Integrated Teat and Evaluation "Test and evaluation shall be integrated throughout the defense acquisition process."	**Support** This fits into the Agile concept of test often and deliver a working product at the end of each iteration.
Enclosure 1 Additional Policy, Section E.1.1.19 Professional Workforce "The Department of Defense shall maintain a fully proficient acquisition, technology, and logistics workforce that is flexible and highly skilled across a range of management, technical, and business disciplines."	**Support** This section provides support for training government personnel in Agile if that is to be used on the program.
Enclosure 1 Additional Policy, Section E.1.1.21 Program Stability "The MDA shall determine the appropriate point at which to fully fund an acquisition program, generally when a system concept and design have been selected,…"	**Challenge** This section discusses developing realistic schedules, investment plans, and affordability assessments. This suggests apriori design for the program which is counter to Agile. However, this may be an educational issue more than an Agile issue to resolve. A lot depends on the level of information needed to make this decision and the type of system being developed.
Enclosure 1 Additional Policy, Section E.1.1.27 Systems Engineering "A modular, open-systems approach shall be employed, where feasible."	**Support** This section requires the acquisition program to be managed using a systems engineering approach that optimizes total system performance and minimizes total ownership costs. In many respects Agile supports this concept and there should not be any issues if the PMO decides to employ Agile.

4.2 Regarding DoDI 5000.02

The barriers to adopting Agile in the DoD appear to be primarily cultural. That is to say that there is little in the way of regulation or guidance provided in DoDI 5000.02 that would prevent the use of Agile. This instruction does impose specific constraints on the acquisition office, but these constraints would be true of any development environment.

On the other hand, the Federal Acquisition Regulations (FARs) impose significant obstacles to collaborative endeavors. In fact, since the system tries to encourage competition (and since the competition must be fair), in many cases users are actually prevented from collaborating with system developers until late in the acquisition life cycle. Further, the mechanisms that are typically imposed by acquisition offices to monitor and control their system developers (such as earned value, or independent cost estimation) are significantly different when the developer is working in an Agile world. As stated earlier, the differences between using Agile and a more traditional method require different management approaches for the advantages of Agile to be fully realized.

4.2.1 Agile Impact to Acquisition: Scenarios

As a basis for discussing how Agile might impact a typical acquisition, let’s look at two very simplified scenarios: a non-Agile software acquisition and an Agile software acquisition. These scenarios primarily relate to an acquisition initiated during the Engineering and Manufacturing Development phase of the acquisition life cycle.

Non-Agile Software Acquisition

In this scenario, a capability document would be created and then a standard government contracting process would be used to select a contractor. The contractor would follow a standard development process that produces the requirements, specifications, and designs that would be reviewed and approved by the acquisition office. Typically, users participate in milestone reviews that would accompany the contractor’s development phases. Once the reviews were complete and the requirements, specifications and designs approved by the acquisition office, the contractor would begin the software implementation. Once the software development is complete, the system would be integrated and go through system and acceptance testing. The first time the acquisition office and users get to try the software out to see if it really works as they want it to is during this testing, which might be many months or even years after the contract was signed.

Agile Software Acquisition

This scenario would start out the same as the non-Agile situation. A capability document would be produced and used as the basis for selecting a contractor. However; from this point on things would be different. Instead of the contractor producing a series of requirements, specifications, and design documents to guide the implementation, the government would provide a “user representative” to the contractor. This user representative would be a member of the development team and would support creation of a set of user stories[27] that describe the user capabilities in terms of simple features (that the users need). The user

[27] This assumes that an overall release plan if needed for the project has already been created. The release plan could be at the feature or capability level.

representative would then collaborate with the development team to prioritize these stories. The team would begin implementing the stories by first selecting stories with the highest user priority and those stories they could implement in a short (typically two- to four-week) iteration. At the end of each iteration, the team would produce a working system that implements all the user stories that have been completed to-date. The acquisition office and users would be able to try the software out at the end of each iteration, providing feedback to the development team. They could even add new user stories, change or delete existing user stories, and reprioritize all the user stories. This process of short iterations continues until all the user stories are completed or until the acquisition office and user agree that the system is good enough; then the system goes through final testing.

Some key differences between these two scenarios are:

- Instead of producing a complete detailed design up front, the Agile team begins with a skeleton architecture/design up front; the architecture evolves over the iterations.
- The Agile process produces a testable system at the end of each iteration that the users can try out (in contrast to non-Agile processes that typically don't provide a user testable system until software development is fully completed).
- The Agile process accepts changes at the start of each iteration. Coupled with the user's evaluation of the testable system at the end of each iteration, this keeps the Agile development team focused on what is most important to the users.

4.2.2 Agile within the Acquisition Life Cycle Phases

The DoDI 5000.02 describes a series of life cycle phases that "establishes a simplified and flexible management framework for translating capability needs and technology opportunities, based on approved capability needs, into stable, affordable, and well-managed acquisition programs that include weapon systems, services, and automated information systems (AISs)." [**5**]

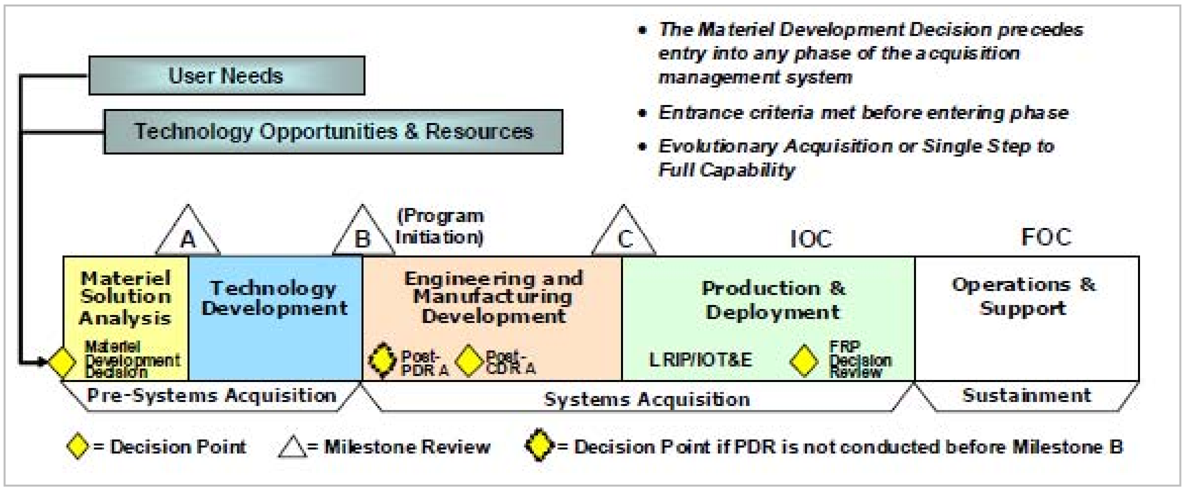

User Needs
Technology Opportunities & Resources
• The Materiel Development Decision precedes entry into any phase of the acquisition management system
• Entrance criteria met before entering phase
• Evolutionary Acquisition or Single Step to Full Capability
A
B
(Program Initiation)
C
IOC
FOC
Materiel Solution Analysis
Materiel Development Decision
Technology Development
Engineering and Manufacturing Development
Post-PDR A
Post-CDR A
Production & Deployment
LRIP/IOT&E
FRP Decision Review
Operations & Support
Pre-Systems Acquisition
Systems Acquisition
Sustainment
= Decision Point
= Milestone Review
= Decision Point if PDR is not conducted before Milestone B

*Figure 5: The Defense Acquisition Management System [**6**]*

Figure 5 shows how acquisition is supposed to work. In brief, there are five life cycle phases, each of which is separated by a milestone of one kind or another. Within each life cycle phase there are opportunities to develop and demonstrate the capability that is desired. As the acquisition progresses through the phases, options for system alternatives are systematically identified and evaluated. As a result of this analysis, some potential solutions survive, while others are eliminated. Eventually, a system configuration that meets stakeholder needs is specified, developed, built, evaluated, deployed, and maintained.

Within this construct there are opportunities for use of Agile. In fact, within almost every life cycle phase, there is opportunity. For example, as discussed above, one of the reasons for using Agile is to establish a validation mechanism for user requirements during the development process. However, specifically in the earlier life cycle stages, such as Material Solution Analysis and/or Technology Development, there is emphasis in the DoDI on “risk reduction.” A managed typical risk would relate to the satisfaction of the Key Performance Parameter (KPP) requirements of the planned system. The reason for this is simple: We want to achieve the best system performance possible, so initially we establish aggressive KPPs. In these earlier life cycle phases, these KPP-related risks might be mitigated by investigating advanced technologies, often requiring the construction of a prototype system or system element. The use of Agile would work well under these circumstances. The frequent builds and evaluations that occur when using Agile should provide a healthy environment for establishing the feasibility of these KPPs.

On the other hand, the instruction does make it clear that a typical entry criterion for progress into the next phase of activity is an estimate of program life-cycle costs. As mentioned earlier, the DoD-sanctioned cost estimation model for developing a system using Agile has not been established. In order for a program to be able to secure funding, they will have to convince their oversight authority that the funding they are requesting is adequate. If the contractor they are using has a significant amount of historical data for validating their cost estimates in the Agile context, this should be less of an issue. Until there is a better industry-wide understanding of the cost estimation methods for Agile resistance to the Agile approach should be expected.[28]

The next life cycle phase described by the DoDI 5000.02 is the Engineering and Manufacturing Development phase. In this phase there will be significant effort associated with software development. In fact, with most software-intensive-systems, this is where the bulk of the software will be produced. This phase is probably where Agile is most likely to be applied. The factors that would influence the decision might involve the length of the development cycle, the “complexity” of the system, the size of the software development team, and the team’s experience with Agile.

It is important to note that the other parts of the system undergo a “system engineering” or “Waterfall” life cycle during this life cycle phase. That is to say that there are formal milestone reviews (such as SRR, PDR, CDR, TRR) to be conducted. It is a challenge for any software development methodology to be consistent with the system’s life cycle. For example, a large system may have an infrastructure software component that is necessary for verification testing of other system components. It is common practice to accelerate the design and build of this component. When this occurs, it would be normal to take the design for this component far beyond the point of “critical design” long before the system’s Critical Design Review (CDR) is to be conducted. The user involvement that is provided by Agile might provide some protection for the system developer under these circumstances. Conversely, it is equally likely that other elements of the software design would not have been designed at the CDR. These software elements are likely to be viewed by the team as “low-risk,” with little or no impact on the system design. However, there is little doubt that the Systems Engineering Waterfall life cycle does not align well with modern

[28] Cost management when using Agile tends to be more of an issue for fixed price contracts where you really have to know what the requirements are up front or the contractor has to take on a lot of risk. For cost plus contracts, this is less of an issue.

software project life cycle approaches. Therefore, it is important to bridge the phase requirements when using Agile to ensure that the reviews don't hinder the ability of the contractor to deliver software increments.

During the next life cycle phase, Production and Deployment, it is rare that a major element of the system cost is software related. From a software development perspective, this phase is quite similar to the next life cycle phase, Operations and Support, where sustainment of the software is conducted. It is assumed that the software previously developed (during the Engineering and Manufacturing Development phase) is mature and stable, so the anticipated software effort expended during this phase is low and should follow a sustainment model, driven by the need to correct errors observed during qualification testing, or providing enhancements as requested by program stakeholders. It is quite possible for a software development team working in these life cycle phases to follow an Agile approach. Quite often the features requested during this phase are modifications that are only relevant within the context of the system that had been previously developed. The aspect of user involvement that naturally occurs at this point of the life cycle makes it easier for the use of a collaborative approach.

It should be noted that some of the Agile methods might not be as practical as others[29] during the Operations & Support phase. For example, it is quite likely that the capability provided during sustainment is planned to be provided over a significant period of time, typically on the order of two years. While the involvement of the user might be beneficial, the frequent releases may not be useful because of limitations with the verification and validation environments required for deployed systems. On the other hand, this constraint should not preclude the use of Agile during this stage of development.

Finally, a detailed analysis was conducted of the DoDI 5000.02 with specific paragraphs extracted, and detailed comments made. The summary of this analysis is provided in Table 2.

[29] Kanban / lean style of Agile might be the most relevant for this phase.

Table 2: Analysis of Acquisition Life Cycle Phases and DoDI 5000.02

Life Cycle Phase	DoDI 5000.02 Guidance (excerpts)	Considerations
Material Solution and Analysis (MSA)	The Analysis of Alternatives (AoA) shall focus on identification and analysis of alternatives, measures of effectiveness, cost, schedule, concepts of operations, and overall risk. The AoA shall assess the critical technology elements (CTEs) associated with each proposed materiel solution, including technology maturity**,** integration risk, manufacturing feasibility, and, where necessary, technology maturation and demonstration needs. To achieve the best possible system solution, emphasis shall be placed on innovation and competition. Existing commercial-off-the-shelf (COTS) functionality and solutions drawn from a diversified range of large and small businesses shall be considered.	Depending on the type of system being acquired, software development may be needed to verify/validate measures of effectiveness. For instance, for many Automated Information Systems (AISs) such as travel or Human Resources, there are no CTEs. For weapons systems, CTEs may have significant capability provided as software. Based on technology maturity and integration risk, there may be software development required for technology demonstration. An Agile rapid prototyping process may be advantageous. Although Agile methods may be desired, competition may preclude users from participating in the software development activity.
Technology Development (TD)	Entrance into this phase depends on the completion of the AoA, a proposed materiel solution, and full funding for planned Technology Development Phase activity.	Agile methods may not mesh with “traditional“ cost estimation and accounting techniques already sanctioned by DoD. “Full funding” in the Agile sense may require a cultural, contractual, and/or legal adjustment and creation of DoD sanctioned Agile cost estimation model.
	The Technology Development Strategy (TDS) shall document the following: (a) The rationale for adopting an evolutionary strategy (the preferred approach) or using a single-step-to-full-capability strategy (e.g., for common supply items or COTS items). For an evolutionary acquisition, the TDS shall include a preliminary description of how the materiel solution will be divided into acquisition increments based on mature technology and an appropriate limitation on the number of prototype units or engineering development models that may be produced in support of a Technology Development Phase; (b) A preliminary acquisition strategy, including overall cost, schedule, and performance goals for the total research and development program; (c) Specific cost, schedule, and performance goals, including exit criteria, for the Technology Development Phase; (h) A summary of the CAIG-approved Cost and Software Data Reporting (CSDR) Plan(s) for the Technology Development Phase (see Section 3 in Enclosure 7).	The emphasis on prototypes and engineering development models should include software. An Agile process should help with prototyping and developing engineering models. For references to cost and schedule see note above.

	The TDS and associated funding shall provide for two or more competing teams producing prototypes of the system and/or key system elements prior to, or through, Milestone B. Prototype systems or appropriate component-level prototyping shall be employed to reduce technical risk, validate designs and cost estimates, evaluate manufacturing processes, and refine requirements. Information technology initiatives shall prototype subsets of overall functionality using one or more teams, with the intention of reducing enterprise architecture risks, prioritizing functionality, and facilitating process redesign.	Although competition may be desired, there must be a way to allow the competition to occur without removing Users from the development process. Under current practices, Users are not allowed to participate in competitive downselects. Surrogates could be considered instead.
	When consistent with technology development phase objectives, associated prototyping activity, and the MDA approved TDS, the PM shall plan a Preliminary Design Review (PDR) before Milestone B. PDR planning shall be reflected in the TDS and shall be conducted for the candidate design(s) to establish the allocated baseline (hardware, software, human/support systems) and underlying architectures and to define a high-confidence design. All system elements (hardware and software) shall be at a level of maturity commensurate with the PDR entrance and exit criteria. A successful PDR will inform requirements trades; improve cost estimation; and identify remaining design, integration, and manufacturing risks. The PDR shall be conducted at the system level and include user representatives and associated certification authorities. The PDR Report shall be provided to the MDA at Milestone B and include recommended requirements trades based upon an assessment of cost, schedule, and performance risk.	Milestones represent a completion event and sign-off for government. Documentation must be prepared by the developer that documents the software design. Those using Agile methods should prepare such documentation as appropriate.
Engineering and Manufacturing Development (EMD)	(b) Post-PDR Assessment. If a PDR has not been conducted prior to Milestone B, the PM shall plan for a PDR as soon as feasible after program initiation. PDR planning shall be reflected in the Acquisition Strategy and conducted consistent with the policies specified in paragraph 5.d.(6). Following PDR, the PM shall plan and the MDA shall conduct a formal Post-PDR Assessment. The PDR report shall be provided to the MDA prior to the assessment and reflect any requirements trades based upon the PM's assessment of cost, schedule, and performance risk. The MDA will consider the results of the PDR and the PM's assessment, and determine whether remedial action is necessary to achieve APB objectives. The results of the MDA's Post-PDR Assessment shall be documented in an ADM.	See Milestone comment (above). In addition, consider that assessment may need to be ongoing and incremental rather than milestone based.
	(c) Post-CDR Assessment. The MDA shall conduct a formal program assessment following system-level CDR. The system-level CDR provides an opportunity to assess design maturity as evidenced by measures such as: successful completion of subsystem CDRs; the percentage of hardware and software product build-to specifications and drawings completed and under configuration management; planned corrective actions to hardware/software deficiencies; adequate developmental testing; an assessment of environment, safety and occupational health risks; a completed failure modes and effects analysis; the identification of key system characteristics; the maturity of critical manufacturing processes; and an estimate of system reliability based on demonstrated reliability rates.	See Milestone comment (above).
	(3) Each program or increment shall have an APB (see Section 4 and Table 6 in Enclosure 4) establishing program goals – thresholds and objectives – for the minimum number of cost, schedule, and performance parameters that describe the program over its life cycle.	Agile methods may not mesh with "traditional" cost estimation and accounting techniques already sanctioned by DoD. "Full funding" in the Agile sense may require a cultural adjustment in the government and creation of DoD sanctioned Agile cost estimation model.

	(4) An affordability determination results from the process of addressing cost during the requirements process and is included in each CDD using life-cycle cost or, if available, total ownership cost. Transition into EMD also requires full funding (i.e., inclusion of the dollars and manpower needed for all current and future efforts to carry out the acquisition strategy in the budget and out-year program), which shall be programmed in anticipation of the Milestone B decision. In general, a Milestone B should be planned when a system concept has been selected, a PM has been assigned, requirements have been approved, and engineering and manufacturing development is ready to begin. In no case shall Milestone B be approved without full funding. The DoD Components shall fully fund their share of approved joint and international cooperative program commitments.	Agile methods may not mesh with “traditional” cost estimation and accounting techniques. “Full funding” in the Agile sense may require a cultural adjustment in the government and creation of DoD sanctioned Agile cost estimation model.
Operations and Support (OS)	(b) Life-cycle sustainment considerations include supply; maintenance; transportation; sustaining engineering; data management; configuration management; HSI; environment, safety (including explosives safety), and occupational health; protection of critical program information and anti-tamper provisions; supportability; and interoperability.	The Agile development process must produce adequate documentation of the design and implementation to enable a different contractor to assume the Operations and Support responsibilities.
	(c) Effective sustainment of systems results from the design and development of reliable and maintainable systems through the continuous application of a robust systems engineering methodology.	See Sustainment note above.

4.3 Foundational Concerns

While policies, regulations, and other governing documents are a large concern for anyone in a PMO thinking about adopting Agile, there are some other underlying concerns that will form the basis for developing the application of Agile. *The most significant of these are culture, training, and customer interaction.*

Culture

Culture is inherent in any organization; in many ways it reflects the methodology being used to develop the product. It is a mindset, a way of thinking and a way of doing business. Culture becomes ingrained into the organization and is usually intertwined with everything the organization does. This includes the organizational structure, the rewards system, the communication style, the decision making style, and the staffing model (types of personnel, roles and responsibilities, team make-up, etc). The Agile culture is quite different from the traditional Waterfall culture. This in itself could be a huge obstacle to the adoption of Agile.

Training

Training is essential when adopting Agile methods: while the constructs and principles seem readily apparent and easy to understand, the actual implementation is more difficult than one would think. Consider that the PMO will be asking people to change habits that have ingrained for years or decades; PMOs will ask them to change the way they do business, conduct their work, and spend their days. Many times the staffing profile of personnel that thrive in an Agile environment is totally opposite to that of those who thrive in a Waterfall environment. People can adapt but this type of fundamental change is not easy. Finding an Agile coach to help the organization move to Agile is essential.

Customer Interaction

One of the key tenets of Agile is access to the customer—the end user; this is essential to the Agile way of doing business. In a government acquisition environment, access to the end user is not always possible. In many cases, there are multiple end users for the product. Staffing this position is problematic due to resource availability, representation of all users, and the type of personnel typically available for this type of interaction.

These concerns need to be addressed by the organization before it begins using Agile. Some ideas on how to do this are provide in Section 5, with common objections to Agile in Appendix B, and areas to consider when embarking on using Agile in Appendix C.

5 Considerations for Applying Agile in the DoD

"Neither agile nor plan-driven methods provide a methodological silver bullet that slays Fred Brooks' software engineering werewolf… Elements of both agile and plan-driven approaches may be characterized as lead bullets." –Barry Boehm and Richard Turner *[**4**]*

For those who are looking to Agile to solve all their software woes, be aware—that particular nirvana will not be presented here. Agile is just another "lead bullet" in the arsenal of methods, practices, techniques, and procedures that can be used to help solve software woes. One statistical study concluded that "little empirical research had been conducted in establishing whether customer satisfaction in the use and results of Agile-driven software development methods was greater than the customer satisfaction in the use and results of plan-driven software development methods." [**7**] This study went on to say that "both methods satisfy their respective customers under a wide range of different situations." Thus, Agile must be applied appropriately and will require discipline.

During the 2009 Agile Development Practices Conference, Alistair Cockburn said during his keynote speech that the concepts of Agile were not new. He went on to say that the concepts of Agile were ones that had been used successfully over the years, and the Agile Manifesto gathered and documented them.

While reviewing multiple references on Agile, we found that indeed, the concepts used in Agile are not new. Some were used as early as the 1950s and through the 60s and 70s, and on into the 80s [**8**]. The Agile Manifesto gathered and documented the ideas and the Agile movement promoted them for the betterment of software development and added value to the end user.

Some might ask "If these concepts are not new, then what's the big deal?" Upon close inspection there are new components (ideas, practices, theories, etc) and new combinations of those new components with "old" components. The explicit value statements used within Agile are also new. In addition, the practice of Agile is new in that it is now becoming more widely employed with demonstrable benefits. From a DoD perspective or that of any large organization for that matter, the paradigm behind using Agile is significantly different than "business as usual." Business as usual tends to be the known as Waterfall or what Boehm includes under a broader definition as a plan-driven method. The mental models for using Agile or Waterfall are very different. For instance, Waterfall says to define all requirements in advance, but Agile says this is impossible and futile because users don't really know all their requirements until they see a system in operation.

To further differentiate between the two paradigms, plan-driven methods' goals are predictability, stability, and high assurance. These can be thought of as *strategic objectives*. On the other hand, Agile goals are rapid value and responsiveness to change. Agile can be thought of as more *tactical objectives* [**4**]. However, this is no reason to prevent using Agile as part of the strategic approach to solving problems.

There is a culture that emerges around any methodology. The culture for plan-driven methods is different from that of Agile. Neither culture is better or worse than the other, just different. For those choosing to move to Agile, the first thing that must be understood is that it won't be "business as usual" and the PMO will need to change its collective mindset, its paradigm, and its culture.

Jim Highsmith, one of the Agile Manifesto signers, has said there will be barriers and impediments as an organization moves to Agile. For large companies, it can be a multi-year transformation. The PMO needs to determine if Agile will be a match for what it wants to do. Is it a strategy for your project, your division, your whole company? The PMO needs to determine how proficient it will be at change—organizational change. [**9**] Be prepared for organizational change management issues.

Since organizational change is always difficult, why would anyone want to embrace Agile? We did not perform an in-depth look at the various statistics on the benefits of Agile. However, the reports and literature about Agile's performance we did look at varied from extremely impressive, which some might think a little too good to be true to moderately good. One example shows that by using Agile, costs decrease from as little as 5 to as much as 61 percent, with schedule decreasing from as little as 24 to as much as 58 percent, and cumulative defects decreasing from as little as 11 to as much as 83 percent [**10**].

Even if one is skeptical and only believes the lower end of these statistics, Agile beckons to be tried to reduce cost and improve benefits and quality for the DoD. Before jumping into the Agile world, take time to consider how Agile can benefit your program, what the issues will be, and if perhaps a hybrid approach (combination of Waterfall and Agile) is the best approach.

Some of the concepts that need to be considered when embarking on the use of Agile are discussed below. The discussion assumes the government will be contracting with a firm to actually do the development. Since the contractor will be creating the organization structure, it is important the government understands what it is and how they interact within that structure. The better the understanding, the less likely there will be inadvertent roadblocks or obstacles created to impede the progress of the Agile team(s). If the government is doing the development internally, some of the actions may differ and would be accomplished by the government. We considered the following concepts.

- Acquisition life cycle
- Team environment to include specific Agile method, team communication, distributed teams, size of program, potential encapsulation
- End-user access
- Training and coaching
- Oversight including milestone reviews, documentation, evaluation (metrics)
- Rewards and incentives
- Team composition
- Culture

Some of the discussion will sound familiar as it parallels feedback we obtained during our program interviews in Section 3. This is not surprising as the concepts were actual issues the programs dealt with during their use of Agile. The concepts discussed here overlap and are intertwined. In many cases, the concepts are mutually reinforcing.

5.1 Acquisition Life Cycle

The acquisition life cycle consists of multiple phases: Materiel Solution Analysis, Technology Development, Engineering and Manufacturing Development, Production & Deployment and Operations & Support. Each of these phases presents unique challenges and opportunities. Some phases lend themselves to the use of Agile better than others. Agile was used on the programs interviewed spanning all the life-cycle phases except the Materiel Solution Analysis phase. However, how Agile was employed varied from program to program. The PMO should determine how to best employ Agile in their program depending on their specific situation. In the following paragraphs, we propose questions to ask and identify issues to consider in building an Agile program. A more in-depth discussion of life cycle phases is provided in Section 4.3.

If the PMO is doing a Request for Proposal (RFP), no matter which phase, ensure that the RFP contains language that allows the use of Agile. In many instances, the traditional RFP language makes it difficult, if not impossible, to propose an Agile-based solution. One consideration is the types of reviews and documents required. If the PMO wants to employ Agile, be prepared to allow for “Agile style” document development, i.e., incremental development of documents and data for reviews that result from the individual iterations and/or releases. This might not seem much different from what the traditional methods provide but consider the level of detail may be sparser using Agile in the earlier versions of the documents. Even final documents might not contain the amount of detail provided in traditional documents. The key here is not the volume, but the content. A necessary and sufficient criterion is that all important data required for operation and maintenance of the system are supplied.

5.2 Team Environment

Earlier in this report, we discussed findings in an area called infrastructure. By infrastructure we mean the structure of the team and the context within which the team operates. Organization structure and environmental support structure both need to be established to support an Agile implementation. The context of a program and its inherent organizational structure are related.

For this report we made the assumption that we were dealing with software only or software-intensive systems. Many systems contain software and could be considered software intensive but the software is only a small part of the overall system and certainly not the end item being procured. For large systems acquiring end items like tanks, ships, planes, or satellites, the Agile software team may need to be encapsulated from the rest of the program.[30] This would entail determining the boundaries or interfaces to the rest of the system and using those as constraints to create the boundary for the Agile software project. These would become constraints for the software development and would be part of any working assumptions for the software environment. For instance, the software could be developed and tested within the Agile environment but then “delivered” to that full system for system test with the tank, missile, ship, etc.

Due to the size and complexity of most DoD programs, multiple iteration teams will be needed. The number is dependent upon the program and in some instances the locations of the contractor team. The larger the number of teams, the more complicated the communications and the greater the need for more users to be involved. In an ideal situation, each iteration team would have access to their own dedicated end user. However, that is not practical in the DoD environment so alternatives need to be employed. Consider the use of proxies, rotating personnel every “x” weeks (x usually is two-four weeks), or perhaps a separate “team” of subject matter experts (SMEs) accessible by the iteration teams as needed.

The structure of the overall program team—especially the contractor team—is dependent upon which Agile method is chosen. Things like pair programming and scrums are just two examples of practices within Agile methods. Typically the contractor determines the “flavor” of Agile. However, the govern-

[30] For other systems, such as AIS, this is usually not an issue.

ment team needs to be responsive and supportive of that method. Otherwise, using Agile will have less than optimal results.

The Agile team also must exhibit behavior reflecting the approach. Leffingwell describes seven practices observed to scale up to enterprise-level development projects, and we have adopted his terminology for this summary treatment [**11**]. A more detailed treatment of these practices is reserved for future work.

The Define/Build/Test Component

Three basic skills are combined in the component team: define, build, and test, operating cooperatively within a pre-defined period, known as a *time box*. The juxtaposition of these skill sets into one team tends to run counter to some conventional methods employed in DoD programs, where these players are often separated by intent.

Two-Level Planning

Two-level planning is portrayed as providing both guidance of how software is to be inserted into the operational environment as well as allowing some flexibility to accommodate what is learned during development:

- The “top level” of the planning cycle is termed release level planning. This cycle of planning defines series of releases that broadly define capability to be contained. This could be done at the feature set level.
- The “second level” of the planning cycle is termed iteration level planning, where iterations break the release into a set of iterations that can be time-boxed.

Mastering the Iteration

The ability of a team to reliably execute a sequence of iterations may well be the key behavior that distinguishes a team capable of exploiting Agile techniques in a large organization. If this capability is not present, the likelihood of success is minimal at best.

The iteration consists of the following key activities in a small time box:

- creation of complete, tested, working code implementing a set of features
- integration of the developed code into the working baseline within the timeframe of the iteration

The result of a given iteration is potentially releasable to the customer.

Producing Smaller and More Frequent Releases

It is clear that one natural effect of the expectations is to desire more frequent feedback from the customer and/or stakeholders to avoid large-scale course corrections. The shorter duration of iterations will help to maintain more or less continuous feedback from the customer. In particular, for feature sets that may evolve due to improved customer understanding of needs, this model of short iterations offers a more timely alternative.

Concurrent Testing

Concurrent testing practices are based upon thorough testing of code both during development and during integration. The goal is that all code is tested. Gamma[31] and others advocate a "test first" developmental approach [**12**] where the unit tests for software are created prior to the actual development. Gamma also advocates frequent use of the unit tests during actual development.

Continuous Integration

Continuous integration may well be the most useful and controversial practice advocated in the Agile community. The continuous integration model diverges from the usual "V-shaped" model advocated by traditional systems engineering practice employed in DoD programs. In the V-shaped model, requirements synthesis, allocation, and development are carried out in a top-down fashion. This is followed by a bottom-up sequence of integration and verification activities, leading to a product ready for use acceptance or sale.

Continuous integration of software is contingent upon the ability to concurrently execute two crucial activities: (1) collect incremental changes from multiple developers on a regular basis, ideally on a daily basis, and (2) perform the "nightly build" discipline, where all changes are brought together in an incremental software baseline, which is in turn compiled and tested with the available unit and regression tests.

Regular Reflection and Adaptation

Reflection and adaptation (sometimes called the retrospective) is the Agile version of 'continuous improvement' that is highlighted in other methodologies. In keeping with the bottom-up discipline of Agile approaches, this introspection is driven down to the individual team level.

5.3 End-User Access

One of the concepts stated in the Agile Manifesto is "Customer Collaboration over contract negotiation." Agile implements this by having continuous contact with the end user. Typically, an end user or his representative is an integral part of the iteration team. This is not always practical in the DoD environment and can be more complicated by the fact that some programs are joint programs involving more than one service. With multiple end users, all with different ideas of what the end product should be, it will be difficult to have a single voice for the end user. Also, the real end user is an operational person who may not have any experience in the acquisition career field, so meeting this Agile requirement is challenging.

Traditional acquisitions try to have user inputs with their success varying depending on availability and a host of other issues. Typically, the acquisition organization speaks for the end user. Thus, they become the proxy for them. In addition, due to contractual rules, only certain people are warranted to talk to the contractors—these are the people who can legally direct the contractor. In Agile, the end user who sits with the iteration team speaks for the program and has the authority to commit. This leads to potential constructive change issues within the DoD arena. It is important to note that no one interviewed expe-

[31] Erich Gamma is a Distinguished Engineer at IBM Rational Software's Zurich lab. He is a coauthor of the first comprehensive book on design patterns [**45**], was a key contributor to the development of the Eclipse software development platform, and led the development of the design patterns employed in the JUnit and related testing infrastructures.

rienced this issue; we mention it here only as a caution and as a potential for changing the contracting officer’s skill set.

Agile in its pure form insists on interactions with the real end user. This interaction will surface end-user disagreements earlier in the project and in the concrete context of demonstrable capability. To overcome this, the PMO and the contractor may have to consider surrogates or proxies. Depending on the PMO’s experience, this use of surrogates may require a culture change—one that may or may not work well.

Another alternative is to use remote collaborative presentation capabilities—as well as wikis, blogs, and live chat, to keep travel costs–from being overwhelming. The challenge for DoD programs is that some of these are not always approved for DoD use, so the program may have to take the lead to get them approved. Another challenge for some organizations is the cultural shift from formal face-to-face review to collaborative, virtual meetings.

5.4 Training and Coaching

Training for Agile is essential. While the concepts of Agile are not new, the specific implementations contain subtleties and nuances that need to be explained. Additional training in the specific contractor method is also a must. Training before starting the project will help to avoid inadvertent roadblocks and prevent some of the more common issues from arising.

Many contractor organizations employ a coach to help them convert their processes to support Agile. A coach and/or an Agile advocate who has “clout” within the PMO is a good addition to the PMO staff. Their presence can answer daily questions, help resolve issues before they become problems and help to ensure the program runs smoothly from an Agile perspective. A word of caution: An Agile advocate or coach without any authority is like not having one at all; they get lost in the chorus of voices demanding to be heard. Keep in mind that the Agile coach for the PMO will have a different role than an Agile coach for the development team. The PMO Agile coach will be there to help the acquisition organization understand what the developers are doing and assist in making both the acquirers and developers work better together.

5.5 Oversight

The existing traditional structure is in place to provide predictability, stability, and high assurance [**4**]. The essence of the traditional structure is created to allow for close oversight and insight into the workings of a program. The structure requires immense amounts of documentation, which is evaluated at key milestones throughout the program. These documents and their review along with the accompanying capstone events (PDR, CDR, etc) provide the government with a high level of “comfort” that the program is progressing the way it should. The traditional Earned Value Method System (EVMS) of measurement is also constructed to provide the government a means to monitor the progress of the program. This system is rigid and monitors progress against the plans in both cost and schedule. These plans are reflected in the Integrated Master Schedule (IMS).

Agile is very flexible and promotes the capability of moving tasks and functions from one iteration to another or even deleting them altogether. This fluid environment is very difficult to track using EVMS as implemented today. The fluid environment also makes it difficult to maintain a current IMS.

An analogy might be useful to understand the kind of oversight expected within Agile. In the military there is something called commander's intent:

> *"Commander's intent describes the desired end state. Your intent statement provides a framework for the operation. It does not tell your soldiers what to do. It does give them the overall picture of what you say the company needs to accomplish to be successful. By making your intent a clear, concise, and focused statement, you greatly increase the chances that your soldiers will continue the mission, even when the operation doesn't go as planned."*[32]

One can think of the overall plan for an Agile program as its intent. If the initial plan doesn't work as thought, then the development team alters the plan with the intent still in mind. Allowing the Agile team to follow the intent without detailed direction is based on trust, collaboration, and relationship building.[33] These ideas are core to Agile.

One often hears that Agile is ad hoc and has no planning. Do not confuse formality with discipline. Agile teams tend to be less formal but are highly disciplined. Combining this with the above discussion means that Agile requires considerable planning if the program is to achieve its objectives. However, more of the planning is done at the mid and low levels of the program versus at the high management level on traditional programs.

The issue of how oversight will function on an Agile-based program must be resolved before you start the program. Both the government and contractor need to agree on the method to be used. EVMS can be used for Agile programs but it requires close coordination between those who monitor the EVMS system and those who maintain it. If a capability or task is swapped out for an equivalent task (equivalent in EVMS value) then this could be used. This becomes labor intensive. Other ways of monitoring Agile programs can be used, such as using completed stories or accrued value to the user as measures.

Documentation within Agile is "just enough" to meet the need and provide continuity for the team. This usually is not sufficient for the capstone reviews. Remember that documents are evolved in increments within Agile and that this will have an impact when the complete document is available. Another documentation challenge in the DoD acquisition environment is maintaining enough documentation of the critical architectural information and decisions so that the knowledge can be effectively transitioned when personnel, military, or contractor, leave the program. Thus, an understanding of the content and when a final version of the documentation will be available needs to be negotiated in advance. Some items that can influence these negotiations include the opportunity cost for how much time is spent on documentation versus creating working software and the impact of a Feature Review of the working software, which provides the PMO with an indication of progress instead of reviewing just the documents.

Capstone events, like CDR and PDR, are also issues in the Agile world. One of the programs interviewed broke the capstone event into smaller IDRs which cumulatively equaled the overall capstone event. Again, this needs to be planned in advance for both technical and programmatic reasons.

[32] http://www.globalsecurity.org/military/library/report/call/call_98-24_ch1.htm

[33] Further discussion of these topics is left to future work.

5.6 Rewards and Incentives

In the traditional Waterfall methodology, typical rewards and incentives are individual based rather than being team based. Contracts, team organizations, and other program structures are developed and interpreted to enforce and enable individual awards. The Agile environment is more team oriented and does not thrive well within the traditional reward structure. While this will not be a large concern for the government unless it is doing internal development, anything the government can do to incent the contractors and support the Agile culture is a major advantage.

The government may want to consider incentives that involve embracing and fostering change and sharing data at the enterprise level. One of the problems with making the cultural shift to Agile is that the right incentives are not in place to foster change. Personnel need to be incented to do significant adoption planning and strategy for the technology shift and related business, legal, and operational aspects. If people are incentivized the right way, they will embrace change.

Reuse and information sharing across the enterprise are important metrics according to Agile. This means that you want to incentivize doing things “for the good of all,” not just for the good of an individual program. The problem is that right now DoD programs are structured to compete with each other. This creates a culture of hoarding knowledge for competitive advantage. The DoD needs to think about how to incentivize collaboration across programs even between competing contractors; this will not be easy and may mean that the DoD might have to think about ways to fund programs other than funneling money to a single program.

5.7 Team Composition

One addition to the typical traditional DoD PMO is an Agile advocate. As described in training and coaching, the advocate is someone who can provide real-time answers for the immediate Agile issue. Another addition to the typical staff is an end-user representative who is empowered to make decisions that are binding for the development. Given the nature of government contracting, care must be given to ensure that this user representative has the legal authority to direct the contractor. We can envision a situation where constructive change could become an issue.

The background of the team members may be slightly different than the norm. An ideal Agile team would consist of experienced, high-performing Agile developers in all positions. It is a proven fact that Agile teams are more successful with more experienced and skilled team members so it is important that the government provide environments that are attractive to high-performing individuals. More experienced personnel would have more skills than just coding. They must be able to work from user stories to design, implement, and test features. The government also needs skilled Agile personnel to review the documentation and understand how the Agile software development approach works. Many traditional PMO teams do not have software representatives experienced with modern software development approaches. That could be more problematic in an Agile environment, where any shortfalls quickly become more visible.

Another challenge is keeping high-performing Agile teams together long enough for them to achieve peak performance. This is a challenge because developers can change at the end of a contractual period of performance. The continuity of an Agile team enhances the “tacit knowledge” of the program and this improves overall performance. One recommendation might be to look at putting key Agile technical developers or technical leads under a separate contract vehicle or hire them to work for the government organization itself.

5.8 Culture

One of the most frequent topics of discussion that was heard at the 2009 Agile Development Practices Conference dealt with culture. People talked about how Agile would not succeed if an organization's culture did not support it.

As we said before, culture is the customary knowledge, beliefs, behavior, and traits displayed by an acquisition organization or contractor. The government is heavily invested in the use of plan-driven methods for acquisition of all equipment and systems, whether they are software intensive or not. As a result, the culture of the DoD acquisition community (and that of long-time DoD contractors, as well) is comfortable with Waterfall and skeptical about the use of Agile. Part of this comfort is how the project management has been trained to manage change. Traditional project managers focus on following the plan with minimal change but the Agile manager focuses on adapting successfully to inevitable changes [**9**]. Since neither Agile nor plan-driven approaches fit every problem, a key to changing the culture is to make it so that it is flexible enough to accommodate both Agile and Waterfall—and anything in between.

In order for the DoD to successfully employ Agile it needs to embrace a culture change. The way it thinks about oversight, documentation, team structure, user interaction with the development team and flexible change must be altered. This is not easy. Changing a culture—any culture—is difficult. It is even harder to change a culture that has strong motivations for control since mission critical and life critical systems are involved. A fear associated with safety or mission critical systems is that Agile does not put enough focus on software engineering practices such as analysis and design necessary to achieve key quality attributes such as performance, security, availability, etc. This can be addressed by the architect and how the architectural requirement stories are prioritized within the team. In addition, some understanding of organizational change management and how groups change will be invaluable. Organizational change discussions are left for future work.

Our research shows that starting small and taking gradual steps into the Agile world would help the transition to the new Agile culture. By starting with a smaller project (thus smaller teams), experience is gained that can be applied to larger programs. In fact, we came across a group within the Air Force that has applied Agile concepts and published their own method. It is called Fast, Inexpensive, Simple, and Tiny (FIST) [**13**]. The FIST method has been successful on several small projects. Recently a FIST Manifesto was created; we include the FIST Manifesto in Appendix E as an example of what is being done at the *grass roots* level within the DoD. Agile in many forms is starting to be employed within the DoD.

6 Conclusion

This small study on the current utilization and future applicability of Agile for software development in DoD acquisitions is meant to whet the appetite of those looking for another tool to solve the ever-present conundrum of obtaining good software quickly and as inexpensively as possible.

Agile methods have existed for many years. In fact, they are based on concepts that have been around for decades, but the methods have not been widely used, especially within the DoD. In recent years, Agile has matured, personnel have become more skilled, and some DoD contractors have started to build internal Agile capabilities and initiated usage on DoD programs.

For the complete novice, we provided a short overview of Agile including the Agile Manifesto and its underlying principles. There are multiple flavors of Agile and we listed several of the more common ones for the reader's edification. Finally, we provided a limited comparison of Waterfall to Agile. This shows some high-level similarities and differences.

Several existing programs that we interviewed and existing literature we read expounded on the benefits and the pitfalls of using Agile. The general consensus is that Agile is another tool to be exploited and this will provide benefit to the DoD in the correct environment.

If this is the case, then why isn't Agile used more? We distilled the information we found from interviews into seven areas that address this question. In each area we looked at the issues and some of the solutions that worked. The seven areas are

- Acquisition
- Knowledge of Agile
- Culture
- Oversight
- End-User Involvement
- Integration and Test
- Infrastructure

A review of the DoD 5000 series showed there are minimal barriers and none of the challenges are show stoppers. A lot of these challenge areas depend on the interpretation. Unfortunately for Agile, most of today's interpretations lean towards the more traditional methods like Waterfall.

Finally, we looked at other concepts that need to be explored before employing Agile in the DoD environment. Most important from the authors' viewpoint are end-user participation and culture. However, in order to employ any aspects of Agile, the DoD organization will have to plan for them, anticipate the changes needed in their environment and business model, and apply the hard work to make the changes a reality.

We acknowledge that this report only begins to explore employing Agile within the DoD. During the course of our research we touched on a lot of topics, many of which need further research. Some of the potential future topics in no order of priority include

- Technology—discussion and explanation of different Agile technical concepts and how they apply within DoD. Considerable literature and courses are already available on specific Agile methods.
- Management—discussion and exploration of governance changes, management style for the Agile PM, and management structure for Agile projects (iteration, release, enterprise)
- Contracts and finance—discussion and exploration of costing and estimation for Agile programs, types of contracts and which works best with Agile, and incentives
- Comparison of methodologies—methods—including plan-driven, Agile, and hybrids—that work best for each type of program,
- Benefits from Agile—discussion about how Agile is viewed within the Agile community using risk and a variation of the cost, schedule, quality triangle
- Organizational change management—discussion of what should be changed to work effectively within an Agile environment, how to go about those instituting changes, etc.
- Culture—definition of the Agile culture, what it relies on, how it is different from existing cultures, and how to bridge the gap

We hope that as more government programs use Agile, more findings, observations, and lessons learned will be shared. After all, that is what the retrospective is all about.

In the future, we want to use an overall retrospective to update and enhance this report.

Appendix A: Examples of Agile Methods

There are many methods that fall under the umbrella of Agile. Some focus on the developer (e.g., XP) and others focus on managerial processes (e.g. Scrum). Most of these approaches are evolving and borrow from each other. Examples of specific Agile methods are listed below.

eXtreme Programming (XP)

"…A software development discipline that organizes people to produce higher quality software more productively.…XP attempts to reduce the cost of change by having multiple short development cycles, rather than one long one. In this doctrine changes are a natural, inescapable and desirable aspect of software development projects, and should be planned for instead of attempting to define a stable set of requirements. Extreme Programming also introduces a number of basic values, principles and practices on top of the agile programming framework."[34] While a stable set of requirements is not defined up front, the overall requirements are defined and refined throughout the program based on user feedback.

Scrum

Scrum is a '"process skeleton' which contains sets of practices and predefined roles. The main roles in Scrum are: (1) the 'ScrumMaster,' who maintains the processes (typically in lieu of a project manager); (2) the 'Product Owner,' who represents the stakeholders; (3) the 'Team,' a cross-functional group of about 7 people who do the actual analysis, design, implementation, testing, etc.

During each "sprint," typically a two to four week period (with the length being decided by the team), the team creates a potentially shippable product increment (for example, working and tested software). The set of features that go into a sprint come from the product "backlog," which is a prioritized set of high level requirements of work to be done. Which backlog items go into the sprint is determined during the sprint planning meeting. During this meeting, the product owner informs the team of the items in the product backlog that he or she wants completed. The team then determines how much of this they can commit to complete during the next sprint. During a sprint, no one is allowed to change the sprint backlog, which means that the requirements are frozen for that sprint. After a sprint is completed, the team demonstrates the use of the software."[35]

Adaptive Software Development (ASD)

"ASD replaces the traditional waterfall cycle with a repeating series of speculate, collaborate, and learn cycles. This dynamic cycle provides for continuous learning and adaptation to the emergent state of the project. The characteristics of an ASD life cycle are that it is mission focused, feature based, iterative, time-boxed, risk driven, and change tolerant."[36]

[34] http://en.wikipedia.org/wiki/Extreme_Programming

[35] http://en.wikipedia.org/wiki/Scrum_%28development%29

[36] http://en.wikipedia.org/wiki/Adaptive_Software_Development

Dynamic Systems Development Method (DSDM)

"As an extension of rapid application development (RAD), DSDM focuses on Information Systems projects that are characterized by tight schedules and budgets. DSDM addresses the most common failures of information systems projects, including exceeding budgets, missing deadlines, and lack of user involvement and top-management commitment. By encouraging the use of RAD, however, careless adoption of DSDM may increase the risk of cutting too many corners. DSDM consists of: (1) Three phases: pre-project phase, project life-cycle phase, and post-project phase; (2) A project life-cycle phase subdivided into 5 stages: feasibility study, business study, functional model iteration, design and build iteration, and implementation."[37]

Crystal

"The Crystal methodology is one of the most lightweight, adaptable approaches to software development. Crystal is actually comprised of a family of methodologies (Crystal Clear, Crystal Yellow, Crystal Orange, etc.) whose unique characteristics are driven by several factors such as team size, system criticality, and project priorities. This Crystal family addresses the realization that each project may require a slightly tailored set of policies, practices, and processes in order to meet the project's unique characteristics. Several of the key tenets of Crystal include teamwork, communication, and simplicity, as well as reflection to frequently adjust and improve the process. Like other agile methodologies, Crystal promotes early, frequent delivery of working software, high user involvement, adaptability, and the removal of bureaucracy or distractions. Alistair Cockburn, the originator of Crystal, has released a book, *"Crystal Clear: A Human-Powered Methodology for Small Teams*."[38]

Feature-Driven Development (FDD)

A model-driven short-iteration process that consists of five basic activities: Develop Overall Model, Build Feature List, Plan By Feature, Design By Feature, and Build By Feature. For accurate state reporting and keeping track of the software development project, milestones that mark the progress made on each feature are defined.[39]

Pragmatic Programming

Pragmatic programming follows the principles of *The Pragmatic Programmer* by Andrew Hunt and David Thomas. It is itself a kind of "umbrella" development approach, since it advocates that the developer not stick to any particular methodology but choose the methods and techniques that work best in the specific environment.

Lean Software Development

The term originated in a book by the same name, *Lean Software Development*, by Mary Poppendieck and Tom Poppendieck. The book presents the traditional Lean principles in a modified form, as well as a set of 22 tools and compares the tools to Agile practices. Lean Software Development

[37] http://en.wikipedia.org/wiki/Dynamic_Systems_Development_Method

[38] http://www.versionone.net/Agile101/Methodologies.asp

[39] http://en.wikipedia.org/wiki/Feature_Driven_Development

promotes seven core principles: eliminate waste, amplify learning, decide as late as possible, deliver as fast as possible, empower the team, build integrity in, and see the whole.

Other software development techniques have been mentioned by writers as belonging to the Agile family of approaches, including Kanban [**14**], Rational Unified Process (RUP), Personal Software Process (PSP), Team Software Process (TSP), and Cleanroom [**4**].

Appendix B: Common Objections to Agile

There are common objections lodged against Agile the authors have encountered in the conduct of their professional activities, many of which are echoed in the relevant literature. The most pertinent such objections and responses are discussed below.

Agile and DoD Software Practices are Incompatible

There is a widespread perception that Agile is in rather stark conflict with DoD software development practices. There are a number of facets to this objection, which we will address in turn.

There is a perception among many that DoD development practices mandate a Waterfall approach to software development. A careful examination of DoDD 5000.01, and DoDI 5000.02 does not support the contention that any of this guidance requires a particular software development methodology. Nothing in the FAR appears to prohibit the use of Agile. The same can be said of the more systems-oriented DoD Architecture Framework (DoDAF). It is fair to observe that a lot of the published examples and the explanations, which are linear descriptions of the processes, can be read to suggest that Waterfall is strongly preferred.

We suspect that a lot of DoD acquisitions take the Waterfall approach rather than some iterative or Agile approach because it is consistent with program skill sets and is perceived as the path of least resistance. Nevertheless, there is no mandate to employ a strict Waterfall development methodology that the authors have been able to find.

Agile is New and Risky

Agile is widely perceived as new and experimental, particularly in DoD circles.

In many respects, Agile may be seen as a codification of things that practitioners have been informally doing for some time. Iteration and experimentation have long been carried out under the umbrella of *trade studies*, which often are constrained by limited resource availability and time boxes. Trade studies, the development of breadboard or prototype elements are simply kinds of controlled experiments used to reduce risk. Agile is simply another way to orchestrate controlled experimentation and address uncertainty and risk (user requirements, operational environment, etc.).

Agile techniques have been rather widely employed in commercial software product development, and much of the literature surveyed offers examples based upon such projects. Agile techniques have been cited in some DoD programs (JMPS, SIAP, classified programs). Given that the literature and practice of Agile as a distinct set of methods spans two decades, it seems inappropriate for DoD software programs to dismiss it as it as new and untried. At the writing of this report, the IEEE is pursuing the creation of standards for Agile development practices, which have yet to be published.

The issue of risk and Agile appears to be a red herring as well. The SEI has a rather unique perspective on the software-related failings of numerous DoD programs. It is fair to suggest that observed program problems and failures include the use of all the major documented software development methodologies and significant variants. With that experience, it is difficult to ascribe incrementally increased risk specifically attributable to the choice of a given software development methodology.[40]

Refactoring Is Incompatible with Stable WBS

One of the primary tactics employed in Agile development projects is refactoring, which is the restructuring of software behavior and structure as development unfolds. The goal of this tactic is to take advantage of improved insight into coupling, cohesion, and maintainability as more software structure is created and evaluated. In most DoD projects the Work Breakdown Structure (WBS) is a key organizing artifact. Some observers have expressed the concern that refactoring could affect the WBS, which would introduce a level of turbulence in the WBS that might be unacceptable.

This objection is not compelling. It is typical practice that the WBS for major systems reflects deliverable physical end items, and software to be developed for a given end item is treated as embedded within the end item. If current systems engineering practice is pursued with regard to establishing external interfaces for physical end items (including the software aspects of the interfaces), refactoring will be confined to the end item, and any WBS elements addressing the software relevant to the end item can be structured to accommodate iterative development techniques. The remaining issue is to allow for refactoring within the software within the scope of the end item. The most straightforward mechanism to achieve this is to structure the software elements of the WBS around operational capabilities, not a functional decomposition that posits specific internal interfaces. A structure of this sort will enable refactoring, and disturbances to the WBS should be affected by software changes only when other considerations come into play as well, such as problems in critical item development.

Unwarranted Impact on Contracting Officer

One difficulty that is relevant to the choice of Agile techniques is its impact upon the program office and the contracting officer that deals directly with software issues.

As has been noted elsewhere, Agile techniques tend to promote project planning and documentation practices that are not within the usual experience base of the program office staff. This is not an insurmountable problem and can be addressed with appropriate training.

The most critical impact on the contracting officer is the need to facilitate the high-tempo user/stakeholder involvement that Agile techniques depend upon to implement more frequent user evaluation than is customary. This model places an uncharacteristic burden upon the program office to make knowledgeable users/stakeholders available on a periodic basis to provide feedback, which in turn places a burden upon the user community to release such people for these assignments. This will ordinarily be a planning, personnel, and inter-organizational issue that will merit the creation of program office to military service entity memoranda of agreement to secure people for these roles. If the program office is operating in a competitive situation, where two or more contractors are pursuing the same work there is a potential risk of (a) not treating the two contractors identically, and/or (b) users or stakeholders inadvertently disclosing contractor proprietary information. The following are some obvious tactics that may be employed; some of these tactics are not new:

[40] Jim Highsmith comments, “I always admonish people that in the end, politics always trumps methodology, any methodology.”

- Contracting officers should be present for all interactions between the contractor and users/stakeholders.
- A program office may form multiple user/stakeholder feedback teams and establish procedural and informational firewalls across those groups.
- Should a single user/stakeholder feedback team be employed in a competitive situation, it is critical it be trained and briefed on organizational conflict of interest policies employed by the program office; it is desirable if the participants have basic knowledge of DoD acquisition conventions (e.g. training in DAU or other acquisition courses).

Interpreting User Feedback as Constructive Change

For reference, *constructive change* is defined as an oral or written act or failure to act by authorized Government official construed by contractor as having same effect as a written change order. Such a change must involve (a) a change in performance beyond minimum contract requirements, and (b) word or deed by government representative which requires contractor effort that is not a necessary part of the contract, and (c) it requires ratification.

Some have expressed concerns that user feedback as employed in Agile may be interpreted as constructive change. While this is plausible, it seems no more problematic than ordinary feedback and guidance offered to contractors during dry runs and formal presentations for major reviews, for example, System Requirements Review (SRR), Preliminary Design Review (PDR), Critical Design Review (CDR), and Test Readiness Review (TRR). As mentioned above, it is important for the program office and the contractor who elects to employ Agile to arrive at a careful agreement upon ground rules. It is important that contractual provisions be incorporated that recognize the role of user/stakeholder feedback that may influence architecture/design evolution in future iterations of development[41]. Users/Stakeholder teams must be educated about the limitations of their role in providing guidance to the contractor. Contracting officers must always be present at such feedback sessions to avoid problems. Feedback that is more than clarification, which does indeed alter the previously agreed scope, should be separately handled by contracting officers. While it is impossible to predict what happens once a program has degenerated to the point that litigation occurs, such measures should reduce the risk expressed by critics in this area.

Agile is Too Hard

Agile techniques are different from conventional practice in many DoD and military service circles. However, to dismiss them as “too hard” strikes the authors as a far too pessimistic stance, in view of the growing body of commercial experience with these methods. There are instances where Agile has been employed on recent DoD programs. In view of the painful and expensive legacy of cost, schedule, and performance problems on software-intensive DoD programs, there seems little to be lost by judiciously employing Agile to potentially improve contract performance.

[41] SEI architecture evaluation teams currently encounter this issue when they elicit feedback in the form of stakeholder scenarios. They are important to the architecture team, yet need to be reconciled with contractual requirements.

Appendix C: Areas for Consideration

The following table provides topics that should be considered when embarking on using Agile. We have summarized our observations and findings; however, this list is not complete and should be used only as a beginning guideline. All programs are different and will have unique requirements and issues to resolve.

Table 3: Areas to Consider for Using Agile in the DoD

Area of Concern	Consideration
Content of RFP	Consider using RFP language that does not preclude the use of Agile. This includes the type of reviews and type and content of documents (CDRLs). At this time the authors do not know of any model Agile RFP language.
Organization structure	Consider the use of a coach or advocate within the PMO to help understand the Agile structure.
End-user involvement	How will you provide access to end users? Is there more than one end-user group? How do you have a single voice for all groups? How often will the end user be available for discussion with the development group? Will end users attend demonstrations? What authority to commit the program will the end user have? Consider using proxies or rotating SMEs or a SME team. Consider a hybrid approach using the best practices of Agile and traditional Waterfall techniques.
Training and coaching	Has pre-award training in Agile been created? …given? Who will be trained? Does the training include which contractual phases will use Agile? Is a coach available to work with the team? What authority does the coach have? Has your program had training on the specific Agile method your contractor is employing?
Oversight including reviews	What type of oversight (e.g., EVMS or an alternative such as stories or “user value”) will be employed? What type of capstone event (e.g. CDR or multiple IDRs) will take place? Do the entrance and exit criteria mesh with Agile approach? How will the IMS be created and maintained? Is the PMO trained to use the contractor’s Agile tracking tool? Is the PMO prepared to relinquish *total* control over how change is managed and which capabilities are developed in the short term versus the long term? Is the PMO aware of the mid- and low-level management focus used with Agile? Is the PMO aware of the potential change in “rules of thumb” for number of developers and code size?
Rewards and incentives	What types of incentives are provided by the program? Does it support the use of Agile or undermine it? Does the type of incentives for both the contractor and PMO support the use of Agile?
Team composition	Have you adjusted to include an Agile advocate who has authority? Have the roles and responsibilities been updated to reflect the use of Agile?
Culture	Are you prepared to encourage, institute, promote and sponsor the culture change and the associated issues? Is someone knowledgeable in organizational change management available to work with your team?
Staffing	Is the PMO aware of the different team composition needed to support an Agile project? What type of Agile knowledgeable PMO staff is needed for your program? Does the government PM have Agile experience? Are more Agile software savvy personnel available to support the program? Consider how you will include integration and test personnel – in particular those involved in system acceptance and operational test.
Acquisition, regulations, policies etc used within phases	Review all application regulations and policies to determine any specific impacts to your program. Tailor items or obtain waivers as needed. In particular pay attention to cost, affordability, and cost realism. Also look for potential constraints from independent operations test agencies, information assurance, information superiority and interoperability. Consider hybrid approach or having an IA expert as part of Agile team.
Integration and test	Determine your approach. Encapsulate the “Agile work” so sell off and final integration and test are traditional. Conduct early and frequent involvement of testing personnel (earlier than for traditional methods). If the team does continuous integration, determine how it may affect your test program. Determine if concurrent testing would conflict with the sometimes mandated separation between test personnel and mission development personnel. Determine if access to the target environment will be given to the software integrators.

Infrastructure	Determine shared assets (if any) for the program. Shared assets could be models, common facilities, agreement on measures, etc used by multiple teams supporting the program. Sometimes these become constraints for Agile teams.
Deliverables	Determine deliverable details for each phase using Agile. Ensure adequate documentation for use in operations and support (sustainment) is provided.
Contracting	Determine any contract changes required to support Agile.
Concept of operations (CONOP)	This still applies for Agile. Determine the influence and context the CONOP provides to the Agile stories. Ensure operational architecture is provided up front.
Cost Estimation	Determine how you will evaluate the costs based on Agile since cost and accounting techniques may be different. Determine what “full funding” means in an Agile sense.
Define/build/test component team	Do you have appropriate staffing to represent the end user? This could be a “product owner” who serves as the proxy for the users.
Two-level planning	Determine the contents of each release and iteration. These are typically done by the contractor (developer) with collaboration from the government, which will need to negotiate the government priorities. This is similar to the planning practices already in place with other methods (rolling wave).
Reflection and adaptation (retrospective)	Be prepared to participate in this activity. It is driven from the individual team not top-level management. Thus, some organization change management and a new business model will need to be employed.

Appendix D: Acronyms

Table 4: Acronyms Used in This Report

ACTDs	**Advanced Concept Technology Demonstrations**
ADM	Acquisition Decision Memorandum
AIS	Automated Information System
AoA	Analysis of Alternatives
APB	Acquisition Program Baseline
ASD	Adaptive Software Development
CAIG	Cost Analysis Improvement Group
CDD	Capability Development Document
CDR	Critical Design Review
CDRL	Contract Data Requirements List
CMMI	Capability Maturity Model Integration
CONOP	Concept of Operations
COTS	Commercial-off-the-Shelf
CSDR	Cost and Software Data Reporting
CTE	Critical Technology Element
DAU	Defense Acquisition University
DoD	Department of Defense
DoDAF	DoD Architecture Framework
DoDD	Department of Defense Directive
DoDI	Department of Defense Instruction
DSDM	Dynamic Systems Development Method
EMD	Engineering and Manufacturing Development
EVMS	Earned Value Management System
FAR	Federal Acquisition Regulation
FDD	Feature Driven Development
FIST	Fast, Inexpensive, Simple, Tiny
HSI	Human Systems Integration
IDR	Interim Design Review
IEEE	Institute of Electrical and Electronics Engineers
IMS	Integrated Master Schedule
JMPS	Joint Mission Planning System
KPP	Key Performance Parameter
MDA	Milestone Decision Authority
MSA	Materiel Solution and Analysis
ORS	Operationally Responsive Space
OS	Operations and Support

OSD	Office of the Secretary of Defense
OTA	Operational Test Agency
PDR	Preliminary Design Review
PMO	Program Management Office
RFP	Request for Proposal
RUP	Rational Unified Process
SAF	Secretary of the Air Force
SEI	Software Engineering Institute
SME	Subject Matter Expert
SOW	Statement of Work
SRR	System Requirements Review
TD	Technology Development
TDS	Technology Development Strategy
TRR	Test Readiness Review
TSP	Team Software Process
VMOC	Virtual Mission Operations Center
WBS	Work Breakdown Structure
XP	eXtreme Programming

Appendix E: FIST Manifesto

THE FIST MANIFESTO (*Fast, Inexpensive, Simple, Tiny)*

System development projects should be done by the smallest possible team of talented people, using a short schedule, a small budget and mature technologies to deliver innovative solutions to urgent needs. This approach is called FIST: Fast, Inexpensive, Simple, Tiny.

Short timelines increase agility and stabilize requirements, technology, budgets and people. Short timelines also force accountability, ownership and learning. To maintain short timelines, a project must also exercise restraint over budgets, complexity and size. Increases to the project's budget, complexity or size inevitably reduce its speed.

Accordingly, the FIST approach advocates the following:

Minimize team size, maximize team talent.

Use schedules and budgets to constrain the design.

Insist on simplicity in organizations, processes and technologies.

Incentivize and reward under-runs.

Requirements must be achievable within short time horizons.

Designs must only include mature technologies.

Documents and meetings must be short. Have as many as necessary, as few as possible.

Delivering useful capabilities is the only measure of success.

FIST Principles

A project leader's influence is inversely proportional to the project's budget and schedule.

Creative constraints foster creativity. Adding time and/or money generally does not improve outcomes.

Fixed funding and floating requirements are better than fixed requirements and floating funding.

Complexity is a cost.

Complexity reduces reliability.

Simplicity scales. Complexity doesn't.

An optimal failure costs a little and teaches a lot. When FIST projects fail, they fail optimally.

Iteration drives learning, discovery and efficiency. FIST is iterative.

Talent trumps process.

Teamwork trumps paperwork.

Leadership trumps management.

Trust trumps oversight.

—*Lt Col Dan Ward, USAF, Maj Gabe Mounce, USAF, J. Simmons, Founder Mach 30 Inc., Deji Badiru, PhD, Air Force Institute of Technology, Rolf C. Smith III, USAF, Lt Col Phil Garrant, USAF, Maj Rhet Turnbull, USAF, Richard A. (Dick) Field, Jr., OASD(HA)/TMA, Cynthia J. Wood, US Corps of Engineers, Christopher R. Paparone, Ph.D. US Army Command and General Staff College, Chris Gunderson, Research Associate Professor of Information Science Naval Postgraduate School PI W2COG and NetCert projects, Andy Nulman, President and CMO of Airborne Mobile Inc., Rolf Smith II, John Palmer, PhD, Rick Brennan, Capt Pete Mastro, USAF*[42]

[42] FIST Manifesto signatories as of March 16, 2010.

References/Bibliography

URLs are valid as of the publication date of this document.

[1] D. F. Rico, "Business Value of Agile Methods, Using ROI & Real Options," 2009. [Online]. www.pmibaltimore.org/events/event_details.php?id=452

[2] H. Glazer, J. Dalton, D. Anderson, M. Konrad, and S. Shrum, "CMMI or Agile: Why Not Embrace Both," Carnegie Mellon University, Software Engineering Institute, Pittsburgh, PA, Technical Report CMU/SEI-2008-TN-003, 2008. [Online]. www.sei.cmu.edu/library/abstracts/reports/08tn003.cfm

[3] A. Cockburn, *Agile Software Development*. Addison-Wesley, 2002. [Online]. http://alistair.cockburn.us/Agile+software+development:+the+people+factor

[4] B. Boehm and R. Turner, *Boehm, Barry and Turner, Richard – Balancing Agility and Discipline – A Guide for the Perplexed*. Addison-Wesley, 2004. [Online]. 0321186125

[5] Department of Defense (DoD), "Department of Defense Instruction (DoODI) 5000.02," Dec. 2008. [Online]. https://acc.dau.mil/CommunityBrowser.aspx?id=37343

[6] Department of Defense (DoD), "Department of Defense Directive (DoODD) 5000.01," Nov. 2007. [Online]. https://acc.dau.mil/CommunityBrowser.aspx?id=37343

[7] D. Buresh, *Customer Satisfaction and Agile Methods, VDM Verlag Dr Muller Actiengesellschaft & Co. KG, 2008*. VDM Verlag Dr. Mueller e.K., 2008.

[8] D. F. Rico, H. H. Sayani, and S. Sone, *What is the ROI of Agile vs Traditional Methods? An analysis of XP, TDD, Pair Programming, and Scrum (Using Real Options), synopsis of The Business Value of Agile Software Methods*. J. Ross Publishing, 2009. [Online]. www.jrosspub.com/Engine/Shopping /catalog.asp?store=&category=&itempage=&item=14200&itemonly=1

[9] J. Highsmith, "Advanced Agile Project Management Seminar," in *Agile Development Practices Conference*, Orlando, FL, 2009.

[10] J. Highsmith, "Beyond Scope, Schedule, and Cost: Measuring Agile Performance," in *Agile Development Practices Conference*, Orlando, FL, 2009.

[11] D. Leffingwell, *Scaling Software Agility: Best Practices for Large Enterprises*. Addison-Wesley, 2008. [Online]. www.informit.com/store/product.aspx?isbn=0321458192

[12] JUnit. JUnit.org Resources for Test Driven Development. [Online]. www.junit.org

[13] D. Ward, C. Quaid, and G. Mounce, *The FIST Handbook*. Beavercreek, OH, Rogue Press Book, 2008.

[14] A. Shalloway, "Kanban: A True Integration of Lean and Agile, Agile Development Practices Conference," in *Agile Development Practices Conference*, Orlando, FL, 2009. [Online]. www.sqe.com/ConferenceArchive/AgileDevPractices2009/ConcurrentThursday.html

[15] K. Beck and B. Boehm, "Agility through Discipline: A Debate," *IEEE Computer*, vol. 36, no. 6, Jun. 2003. [Online]. www.computer.org/portal/web/csdl/doi/10.1109/MC.2003.1204374

[16] B. Boehm, "Get Ready for Agile Methods, With Care," *IEEE Computer*, Jan. 2002.

[17] B. Boehm and R. Turner, "Management Challenges to Implementing Agile Processes in Traditional Development Organizations," *IEEE Software*, vol. 22, no. 5, Sep. 2005. [Online]. http://portal.acm.org/citation.cfm?id=1092725

[18] T. DeMarco and B. Boehm, "The Agile Methods Fray," *IEEE Computer*, vol. 35, no. 6, Jun. 2002. [Online]. http://portal.acm.org/citation.cfm?id=622005

[19] J. Highsmith, *Agile Project Management: Creating Innovative Products*. Addison Wesley. [Online]. www.informit.com/store/product.aspx?isbn=0321219775

[20] J. H. Dobbins, "Agile Acquisition Within the Current Policy Framework," in *21st Annual Systems and Software Technology Conference 2009: Technology: Advancing Precision*, Salt Lake City, UT, 2009.

[21] W. P. O'Brien, "Agile Integration of Complex Systems," in *21st Annual Systems and Software Technology Conference 2009: Technology: Advancing Precision*, Salt Lake City, UT, 2009.

[22] P. J. Solomon, "Agile Methods with Performance-Based Earned Value," in *21st Annual Systems and Software Technology Conference 2009: Technology: Advancing Precision*, Salt Lake City, UT, 2009.

[23] K. A. Cianci, "Agile System Development," in *21st Annual Systems and Software Technology Conference 2009: Technology: Advancing Precision*, 2009.

[24] J. O. Clark and S. Johnson, "Agile Systems Engineering and Software Engineering," in *21st Annual Systems and Software Technology Conference 2009: Technology: Advancing Precision*, Salt Lake City, UT, 2009.

[25] E. W. Bingue and D. A. Cook, "The Art of Applying Commercial Best Practices in the DoD," in *21st Annual Systems and Software Technology Conference 2009: Technology: Advancing Precision*, Salt Lake City, UT, 2009.

[26] R. Turner, "Evaluating the Effectiveness of Systems and Software Engineering Methods, Processes and Tools for Use in Defense Programs," in *21st Annual Systems and Software Technology Conference 2009: Technology: Advancing Precision*, Salt Lake City, UT, 2009.

[27] M. R. Coats and H. Koehnemann, "Experiences Applying Agile Practices to Large Systems Development," in *21st Annual Systems and Software Technology Conference (STSC) 2009: Technology: Advancing Precision*, Salt Lake City, UT, 20090.

[28] M. Dwyer, "Updating 'Software Engineering' for the 21st Century," in *21st Annual Systems and Software Technology Conference (STSC) 2009: Technology: Advancing Precision*, Salt Lake City, UT, 2009.

[29] E. Derby, "Collaboration Skills for Agile Teams," *Crosstalk*, Apr. 2007. [Online]. www.stsc.hill.af.mil/Crosstalk/2007/04/0704Derby.html

[30] R. Turner, "Toward Agile Systems Engineering Processes," *Crossalk*, Apr. 2007. [Online]. www.stsc.hill.af.mil/CrossTalk/2007/04/0704Turner.html

[31] G. Miller, "Agile Software Development For The Entire Project," *Crosstalk*, Dec. 2005. [Online]. www.stsc.hill.af.mil/crossTalk/2005/12/0512Miller.html

[32] D. Ward and C. Quaid, "FIST, Part 5, Putting the Pieces Together," *Defense AT&L*, May 2006.

[33] J. Scumniotales, J. McKenna, and P. Egan. (2009, Jul.) Why Scrum Isn’t Enough for Agile Success,. [Online]. Serena Software www.agilejournal.com/news-a-events/events/details/15-webcast-why-scrum-isnt-enough-for-agile-success

[34] (2009, Aug.) Selling Agile, Sliger Consulting, Inc and Computer Aid, Inc (CAI).

[35] (2009, Aug.) An Agile Developer’s Guide to Lean Software Development. [Online]. Computer Aid, Inc (CAI)

[36] J. Highsmith. (2009, Sep.) Beyond Scope, Schedule, and Cost: Measuring Agile Performance. [Online]. http://blog.cutter.com/2009/08/10/beyond-scope-schedule-and-cost-measuring-agile-performance/

[37] "Comparing Extreme Programming, Scrum, and Lean Software Development in Agile," Serena Software, Oct. 2009.

[38] J. Patton, "Using Kanban Techniques to Control Incremental Development," in *21st Annual Systems and Software Technology Conference (STSC) 2009: Technology: Advancing Precision*, Salt Lake City, UT, 2009.

[39] J. E. Lascano, "eXtreme Programming (XP) May Be Embedded Inside Scrum," in *21st Annual Systems and Software Technology Conference (STSC) 2009: Technology: Advancing Precision*, Salt Lake City, UT, 2009.

[40] M. Daconta, "6 trends government IT managers should be wary of," *Federal Computer Week*, Aug. 2009. [Online]. http://fcw.com/articles/2009/08/10/reality-check-it-fads-not-fit-for-government.aspx

[41] Rally Software Development, Inc. (2009) Rally Software Development, Inc.. [Online]. http://www.rallydev.com/downloads/document/161-iteration-planning-guide.html

[42] G. B. Alleman, M. Henderson, and R. Seggelke, "Making Agile Development Work in a Government Contracting Environment: Measuring Velocity with Earned Value," in *Agile Development, June 25-28, 2003,*, Salt Lake City, UT, 2003. [Online]. http://www.informatik.uni-trier.de/~ley/db/conf/agiledc/agiledc2003.html

[43] T. Dybå and T. Dingsøyr, "What Do We Know About Agile Software Development," *IEEE Software*, vol. September/October, 2009.

[44] A. Cockburn, "What Engineering Has in Common With Manufacturing and Why It Matters," *Crosstalk*, Apr. 2007. [Online]. http://www.stsc.hill.af.mil/crosstalk/2007/04/0704Cockburn.html

[45] E. Gamma, R. Helm, R. Johnson, and J. M. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison Wesley, 1994. [Online]. www.informit.com/store/product.aspx?isbn=0201634988

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.			
1. **AGENCY USE ONLY** (Leave Blank)	2. **REPORT DATE** April 2010		3. **REPORT TYPE AND DATES COVERED** Final
4. **TITLE AND SUBTITLE** Considerations for Using Agile in DoD Acquisition			5. **FUNDING NUMBERS** FA8721-05-C-0003
6. **AUTHOR(S)** Mary Ann Lapham, Ray Williams, Charles (Bud) Hammons, Daniel Burton, & Alfred Schenker			
7. **PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)** Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA 15213			8. **PERFORMING ORGANIZATION REPORT NUMBER** CMU/SEI-2010-TN-002
9. **SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)** HQ ESC/XPK, 5 Eglin Street, Hanscom AFB, MA 01731-2116			10. **SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. **SUPPLEMENTARY NOTES**			
12A **DISTRIBUTION/AVAILABILITY STATEMENT** Unclassified/Unlimited, DTIC, NTIS			12B **DISTRIBUTION CODE**
13. **ABSTRACT (MAXIMUM 200 WORDS)** This report explores the questions: Can Agile be used in the DoD environment? If so, how? Lessons learned from actual DoD programs that have employed and are employing Agile are provided as well as information gleaned from the myriad articles and books available on Agile. While this report does not pretend to cover every paper or thought published about Agile in the DoD world, it provides an overview of some challenges in using Agile; an overview of how some programs have addressed these challenges; and some additional recommendations on dealing with these challenges. The intended audience is policy makers, program office staff, and software development contractors who are contemplating proposing the use of Agile software development methods. It is the hope of the authors that this paper stimulates discussion about and appropriate adoption of Agile in the DoD world. We hope to obtain further data so that our list of considerations can be updated and expanded for use by all practitioners.			
14. **SUBJECT TERMS** acquisition, Agile methods, lessons learned, software development, DoD, Department of Defense			15. **NUMBER OF PAGES** 83
16. **PRICE CODE**			
17. **SECURITY CLASSIFICATION OF REPORT** Unclassified	18. **SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	19. **SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	20. **LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. Z39-18
298-102